ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ανάπτυξη Εφαρμογής Διαχείρισης Εξόδων σε Android
Development of an Expenses Manager Android Application

Διπλωματική Εργασία

Γεωργία Τσάκα

Επιβλέπουσα
Τσομπανοπούλου Παναγιώτα
Αναπληρώτρια Καθηγήτρια

Συνεπιβλέπων
Βασιλακόπουλος Μιχαήλ
Αναπληρωτής Καθηγητής

Συνεπιβλέπων
Τσουκαλάς Ελευθέριος
Καθηγητής

Βόλος, Φεβρουάριος 2020

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την κυρία Τσομπανοπούλου που με καθοδήγησε κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.
Επίσης, ευχαριστώ τους γονείς μου για τις προσπάθειες που κατέβαλαν για να με στηρίξουν κατά τη διάρκεια των σπουδών μου και να με κάνουν τον άνθρωπο που είμαι σήμερα.
Τέλος, ευχαριστώ την αδερφή μου που με καταλαβαίνει όσο κανείς και που με έκανε θεία του πιο γοητευτικού μωρού.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Η Δηλούσα

Τσάκα Γεωργία

Περίληψη

Οι εφαρμογές Android έχουν μπει στην καθημερινότητα των ανθρώπων και πλέον προσφέρουν εκτός από διασκέδαση, πολλές χρήσιμες υπηρεσίες και διευκολύνσεις. Μια από αυτές, είναι η καταγραφή των εσόδων και εξόδων που κάνει ένας χρήστης με σκοπό να μπορεί να διαχειριστεί με τον βέλτιστο δυνατό τρόπο τις οικονομικές του δαπάνες. Η παρούσα διπλωματική εργασία πραγματεύεται τη δημιουργία μια εφαρμογής αυτού του περιεχομένου, χρησιμοποιώντας το Android Studio ως περιβάλλον προγραμματισμού, αφού πρώτα γίνει αναφορά σε ήδη υπάρχουσες εφαρμογές με αντίστοιχη λειτουργία. Επιπλέον, αναλύεται η χρήση σύγχρονων εργαλείων, όπως η βάση δεδομένων του Firebase και το Git, το οποίο διευκολύνει τον έλεγχο του αλλαγμένου κώδικα και σταδιακά βοηθάει στην καταγραφή της προόδου της ανάπτυξης της εφαρμογής. Ταυτόχρονα, παρουσιάζονται τα μέσα που προσφέρει το Android για τη δημιουργία μιας εμφανίσιμη και εύχρηστης διεπαφής για τον χρήστη, καθώς επίσης κομμάτια κώδικα, γραμμένα σε Java, που εξηγούν, μεταξύ άλλων, τον τρόπο σύνδεσης και επικοινωνίας της εφαρμογής με το cloud, όπου αποθηκεύονται τα δεδομένα. Τέλος, παρατίθενται τα πλεονεκτήματα χρήσης της συγκεκριμένης εφαρμογής, τα τυχόν μειονεκτήματα και προτείνονται μετέπειτα βελτιώσεις της.

Abstract

Android applications have entered the everyday lives of people, offering not only entertainment, but also useful services and facilitations. One of them is the tracing of all the expenses that a user makes, so as to have control on where their income is spent. This thesis describes the creation of an android application of the aforementioned content, using the Android Studio as a development environment, also mentioning other similar applications already on the market. Moreover, we analyze the use of modern tools, such as the Realtime Database of Firebase as well as Git, which facilitates the control of changed parts of code and is important in recording the development progress of an application. In addition to this, we present the tools and widgets offered by Android, which assist in creating an engaging and convenient application, alongside parts of code written in Java, which represent the steps to connect and communicate with the database, among others. Finally, we evaluate the pros and cons of the application and suggest future ameliorations.

Πίνακας περιεχομένων

Πίνακας εικόνων

1 Εισαγωγή

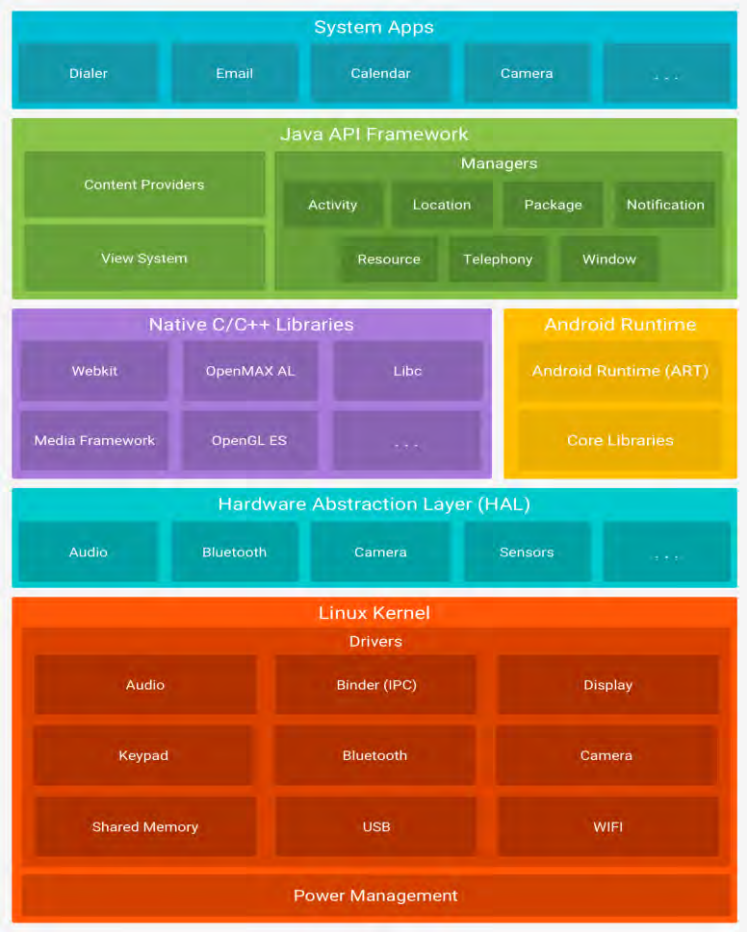
Οι εφαρμογές για Android αρχικά είχαν ως σκοπό την παροχή υπηρεσιών, όπως ανταλλαγή email ή μηνυμάτων, ωστόσο σταδιακά οι απαιτήσεις των χρηστών αυξήθηκαν και επομένως εμφανίστηκε μεγαλύτερη ποικιλία στο είδος των εφαρμογών, όπως παιχνίδια, υπηρεσίες που χρησιμοποιούν την τοποθεσία, αγορά εισιτηρίων κ.α. Πλέον υπάρχουν εκατομμύρια εφαρμογές στο Google Play Store, την επίσημη πλατφόρμα διανομής εφαρμογών της Google, και διατίθενται είτε δωρεάν είτε επί πληρωμή.

1.1 Τι είναι το Android

Το Android είναι το λειτουργικό σύστημα για κινητά τηλέφωνα της Google, το οποίο είναι βασισμένο στον πυρήνα των Linux καθώς και άλλων open source λογισμικών. Το πρώτο κινητό με ενσωματωμένο το Android OS κυκλοφόρησε το 2008 ενώ από το 2011 είναι το κορυφαίο λογισμικό σε πωλήσεις παγκοσμίως με πάνω από δύο δισεκατομμύρια ενεργούς χρήστες. [1]

Η Εικόνα 1 παριστάνει τη δομή του Android λειτουργικού, ξεκινώντας από τον πυρήνα του και καταλήγοντας στις εφαρμογές συστήματος.

1.1.1 Δομή



Εικόνα 1. Η στοίβα λογισμικού Android [2]

Linux Kernel

Ο πυρήνας του Android είναι βασισμένος στον πυρήνα των Linux, καθώς το συγκεκριμένο ευρέως διαδεδομένο open source λογισμικό έχει πολλά πλεονεκτήματα όπως φορητότητα, ασφάλεια και εξαιρετική διαχείριση μνήμης και διεργασιών. [2]

Hardware Abstraction Layer (HAL)

Το HAL αποτελείται από διάφορες βιβλιοθήκες που σχετίζονται με κάποια hardware μονάδα, όπως για παράδειγμα η κάμερα. Είναι, επομένως, ο διαμεσολαβητής μεταξύ του API της Java και του υλικού της συσκευής.

Android Runtime (ART)

Το ART είναι το περιβάλλον που χρησιμοποιεί το Android για να διαχειριστεί το χρόνο που θα τρέξει μια εφαρμογή παίρνοντας τη θέση του Dalvik, το περιβάλλον που χρησιμοποιούνταν

αρχικά για τη διαχείριση χρόνου στα Android projects. Ένα από τα σημαντικότερα χαρακτηριστικά του ART είναι το ahead-of-time (AOT) και just-in-time (JIT) compilation, μια μέθοδος που λαμβάνει σαν είσοδο ένα αρχείο .dex (Dalvik Executable format) και δημιουργεί ένα συμβατό με τη συσκευή εκτελέσιμο αρχείο στο χρόνο εγκατάστασης της εφαρμογής, βελτιώνοντας έτσι την απόδοσή της. Ειδικότερα για API level 28 και πάνω, τα αρχεία .dex συμπιέζονται ακόμη περισσότερο και επομένως η εφαρμογή ανοίγει γρηγορότερα και καταναλώνει λιγότερη μνήμη ευνοώντας έτσι κατώτερης ποιότητας συσκευές. Παράλληλα, το ART συμβάλλει στη μείωση του φαινομένου που είναι γνωστό ως garbage collection, της μνήμης δηλαδή που παραμένει δεσμευμένη χωρίς να χρησιμοποιείται. Πιο συγκεκριμένα, όταν ένα κομμάτι μνήμης γεμίσει και απαιτείται περισσότερος χώρος, συμβαίνει ένα garbage collection pause στο οποίο όλες οι διεργασίες αναστέλλονται και διαγράφονται όλα τα περιττά δεδομένα. Με το ART, λοιπόν, πραγματοποιείται μία σε αντίθεση με τις δύο παύσεις του garbage collector (GC) του Dalvik. Επιπλέον, ο GC του ART ξοδεύει λιγότερο χρόνο να αποδεσμεύσει πρόσφατα δεσμευμένη μνήμη, συλλέγει ταυτόχρονα απορρίμματα εγκαίρως και ελαχιστοποιεί τον κατακερματισμό και την χρήση μνήμης στο παρασκήνιο. Ταυτόχρονα, επιτρέπει την ταυτόχρονη εκτέλεση διεργασιών, με αποτέλεσμα να μην είναι υποχρεωτικό να σταματήσει η εφαρμογή όταν δεν υπάρχει άλλη διαθέσιμη μνήμη. [3] Τέλος, το ART προσφέρει περισσότερες επιλογές για debugging, όπως αναλυτικά διαγνωστικά exceptions και η δυνατότητα να ανασταλεί η λειτουργία του προγράμματος όταν χρησιμοποιηθεί ή αλλάξει ένα συγκεκριμένο πεδίο.

Native C/C++ Libraries

Το ART, το HAL καθώς και άλλα βασικά δομικά στοιχεία του Android χρησιμοποιούν βιβλιοθήκες της C και της C++. Κάποιες από αυτές μπορούν να χρησιμοποιηθούν μέσω του συνόλου εργαλείων Android NDK, για τη δημιουργία εφαρμογών που περιέχουν native code, δηλαδή κώδικα που αλλάζει ανάλογα με τον επεξεργαστή κάθε συσκευής. [4]

Java API Framework

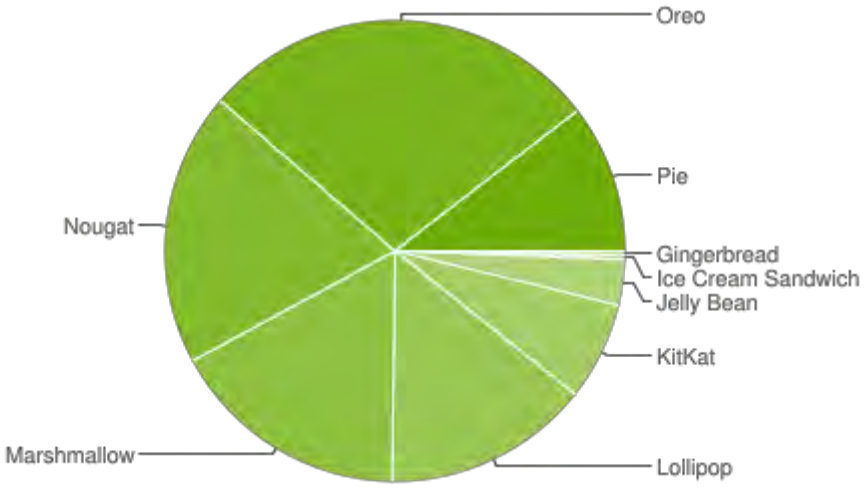
Κάθε εφαρμογή Android περιλαμβάνει ολοκληρωμένα στοιχεία και υπηρεσίες τα οποία παρέχονται από APIs γραμμένα σε Java. Για παράδειγμα, προσφέρονται πλούσια και επεκτάσιμα εργαλεία για τη δημιουργία του user interface (UI) της εφαρμογής, που μπορεί να περιέχει κουμπιά, λίστες, πεδία κειμένου και άλλα. Άλλες παροχές είναι η δυνατότητα εμφάνισης γραφικών, κειμένων και άλλων πόρων που δεν αποτελούνται από κώδικα, η διαχείριση της διάρκειας ζωής της εφαρμογής και η δυνατότητα πρόσβασης σε δεδομένα άλλων εφαρμογών.

System Apps

Τα κινητά τηλέφωνα με Android έχουν κάποιες σταθερές εφαρμογές για ανταλλαγή SMS ή email, ημερολόγια, επαφές κλπ. χωρίς ωστόσο αυτό να απαγορεύει στο χρήστη να εγκαταστήσει εφαρμογές τρίτων για να εκτελέσει τις ίδιες λειτουργίες. Αυτές οι εφαρμογές μπορούν με την ίδια αποτελεσματικότητα να χρησιμοποιηθούν και από τους προγραμματιστές με σκοπό να αποφύγουν να χτίσουν στην εφαρμογή τους μια λειτουργικότητα η οποία παρέχεται από μια ήδη υπάρχουσα [2].

1.1.2 Εκδόσεις λογισμικού

Παρακάτω παρατίθενται όλες οι εκδόσεις του Android OS με χρονολογική σειρά. Επιπλέον, παρουσιάζεται η ποσοστιαία χρήση συσκευών με την αντίστοιχη έκδοση. Επί του παρόντος, η πιο διαδεδομένη είναι η έκδοση Oreo [5].



Εικόνα 2. Στοιχεία που συλλέχθηκαν στο διάστημα 1-7 Μαΐου 2019 [5]

Πίνακας 1. Σχετικός αριθμός συσκευών που χρησιμοποιούνται ανά έκδοση λογισμικού Android [5]

Version	Codename	API	Distribution
2.3.3 - 2.3.7	**Gingerbread**	10	0.3%
4.0.3 - 4.0.4	**Ice Cream Sandwich**	15	0.3%
4.1.x	**Jelly Bean**	16	1.2%
4.2.x		17	1.5%

4.3		18	0.5%
4.4	**KitKat**	19	6.9%
5.0	**Lollipop**	21	3.0%
5.1		22	11.5%
6.0	**Marshmallow**	23	16.9%
7.0	**Nougat**	24	11.4%
7.1		25	7.8%
8.0	**Oreo**	26	12.9%
8.1		27	15.4%
9	**Pie**	28	10.4%

Είναι εμφανές ότι το μεγαλύτερο ποσοστό των χρηστών δε χρησιμοποιεί συσκευές με το πιο ενημερωμένο λειτουργικό σύστημα. Αυτό σημαίνει ότι αν και το πιο πρόσφατο API προσφέρει περισσότερες λειτουργίες, η εφαρμογή πρέπει να υποστηρίζεται και από χαμηλότερα APIs, έτσι ώστε να μπορεί να τη χρησιμοποιήσει περίπου το 90% των χρηστών. Στο αρχείο AndroidManifest.xml που υπάρχει στο Android project κάθε εφαρμογής και περιέχει βασικές πληροφορίες για αυτή, αναφέρονται δύο attributes, το minSdkVersion και το targetSdkVersion, τα οποία θέτουν το χαμηλότερο και υψηλότερο συμβατό με την εφαρμογή API αντίστοιχα. Σε περίπτωση, επομένως που ο προγραμματιστής χρησιμοποιεί κώδικα που εξαρτάται από πιο σύγχρονες εκδόσεις του Android λογισμικού, πρέπει να φροντίσει ότι αυτός τρέχει μόνο όταν το σύστημα είναι συμβατό με τα αντίστοιχα APIs. [6]

1.1.3 Όροι - Κλειδιά

Ξεκινώντας ένα project σε Android, ο προγραμματιστής ορίζει κάποιες παραμετροποιήσεις σε ένα αρχείο xml και δημιουργεί κλάσεις οι οποίες αποτελούν υποκλάσεις κάποιων βασικών

δομών που προσφέρει το Android. Παρακάτω παρατίθενται οι σημαντικότερες από αυτές τις δομές, οι οποίες χρησιμοποιήθηκαν και για την ανάπτυξη της εφαρμογής.

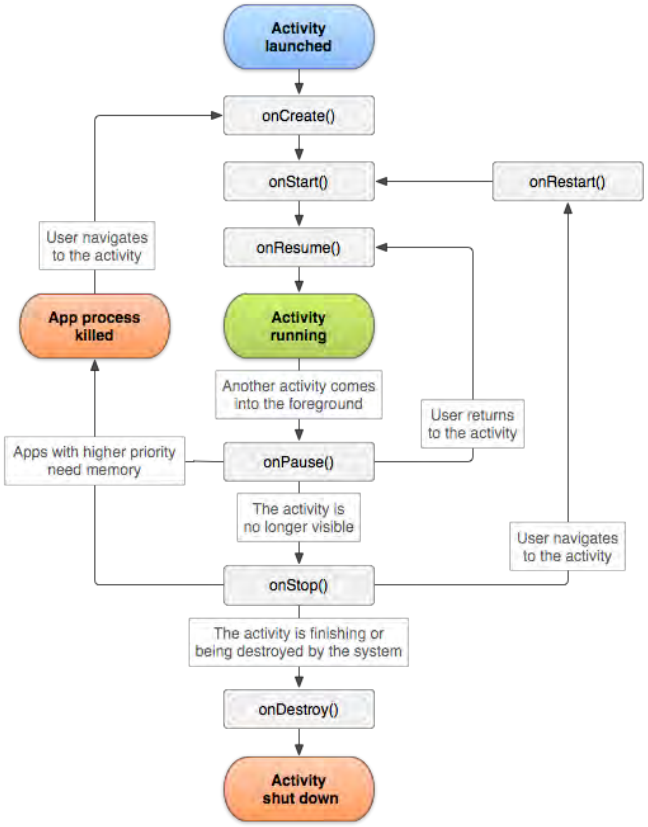
Activities

Η κλάση Activity αποτελεί ένα θεμελιώδες κομμάτι μια εφαρμογής Android, καθώς είναι ο βασικός τρόπος αλληλεπίδρασης με το χρήστη. Συνήθως είναι μία οθόνη της εφαρμογής, επομένως εφόσον μια εφαρμογή έχει πολλές οθόνες, έχει και πολλά activities. Ως κανόνας, μία από αυτά τα activities ορίζεται ως η main activity, η οποία αποτελεί την πρώτη οθόνη που βλέπει ο χρήστης όταν ανοίγει την εφαρμογή. [7]

Κύκλος ζωής του Activity

Κατά την περιήγηση του χρήστη στην εφαρμογή, το στιγμιότυπο ενός Activity μεταβαίνει σε διαφορετικές καταστάσεις και είναι δυνατό να οριστεί η συμπεριφορά του Activity σε περίπτωση αποχώρησης ή επανάκλησης του από το χρήστη. Η σωστή υλοποίηση των μεθόδων που σχετίζονται με τον κύκλο ζωής ενός Activity, των λεγόμενων callbacks, μειώνει την πιθανότητα απροσδόκητου τερματισμού της εφαρμογής σε περίπτωση που ο χρήστης λάβει μία κλήση ή περιστρέψει την οθόνη του, περιορίζει την κατανάλωση πόρων σε στιγμές αδράνειας κ.α.

Για τη μετάβαση από το ένα στάδιο του κύκλου ζωής ενός activity στο άλλο, χρησιμοποιούνται τα παρακάτω έξι θεμελιώδη callbacks.



Εικόνα 3. Μια απλοποιημένη απεικόνιση του κύκλου ζωής του Activity [7]

1. **onCreate**

   Αυτό το callback καλείται όταν δημιουργείται το activity. Εκεί ακολουθείται μια λογική αρχικοποίησης η οποία πρέπει για συμβαίνει μόνο μία φορά κατά τη διάρκεια ζωής του activity. Η μέθοδος παίρνει ως παράμετρο ένα Bundle object, το savedInstanceState, που περιέχει την προηγούμενη αποθηκευμένη κατάσταση του activity αν υπάρχει, διαφορετικά είναι null.

2. **onStart**

   Με τη βοήθεια αυτής της μεθόδου, το activity εμφανίζεται στο προσκήνιο, γίνεται ορατό στο χρήστη και είναι έτοιμο για αλληλεπίδραση. Αμέσως μετά την κλήση της, το activity βγαίνει από το στάδιο εκκίνησης και μεταβαίνει στο στάδιο συνέχισης.

3. **onResume**

   Πλέον το activity βρίσκεται στο στάδιο συνέχισης, όπου ο χρήστης αλληλοεπιδρά με αυτό, και το σύστημα καλεί την onResume στην οποία γίνονται αρχικοποιήσεις που πρέπει να εφαρμόζονται κάθε φορά που το activity μπαίνει στο στάδιο συνέχισης. Το στάδιο αυτό παραμένει έως ότου προκληθεί κάποιο γεγονός που διακόπτει το activity, όπως μια εισερχόμενη κλήση, το σβήσιμο της οθόνης ή η πλοήγηση σε άλλο activity.

4. **onPause**

   Η μέθοδος onPause καλείται όταν υποδεικνύεται ότι το activity δε χρειάζεται πλέον να βρίσκεται στο προσκήνιο και μπορεί να μεταβεί στο στάδιο παύσης. Σε αυτή τη μέθοδο καθορίζονται οι λειτουργίες που πρέπει να σταματήσουν ή να συνεχίσουν να είναι ενεργές με κάποιες τροποποιήσεις. Το στάδιο παύσης προκύπτει σε περιπτώσεις όπως αυτές που περιεγράφηκαν στη μέθοδο onResume, όταν ανοίγει ένα άλλο

ημιδιαφανές activity, δηλαδή ένα activity που δεν καλύπτει ολόκληρη την επιφάνεια της οθόνης, όπως ένας διάλογος, αλλά το εν λόγω activity παραμένει μερικώς ορατό κ.α. Η εκτέλεση της onPause είναι πολύ σύντομη και γι' αυτό δε θα έπρεπε να χρησιμοποιείται για ενέργειες όπως αποθήκευση δεδομένων, κλήσεις δικτύου ή επικοινωνία με τη βάση δεδομένων καθώς πιθανότατα να μην προλάβουν να εκτελεστούν πριν την ολοκλήρωση της μεθόδου. Θα έπρεπε, τέλος, να αναφερθεί πως μετά το πέρας της onPause το activity παραμένει στο στάδιο παύσης έως ότου να συνεχιστεί ή να σταματήσει εντελώς.

5. **onStop**

   Όταν το activity δεν είναι πλέον ορατό στο χρήστη ή έχει ολοκληρώσει τη λειτουργία του και είναι έτοιμο να τερματιστεί, μπαίνει στην κατάσταση διακοπής. Σε αυτό το στάδιο, καλείται η onStop, με σκοπό να απελευθερωθούν οι πόροι που δε χρειάζονται πλέον και να εκτελεστούν λειτουργίες που απαιτούν χρόνο και επεξεργασία, όπως η αποθήκευση πληροφορίας στη βάση δεδομένων. Από το στάδιο διακοπής, το activity μπορεί είτε να αλληλοεπιδράσει εκ νέου με το χρήστη είτε να τερματίσει εντελώς. Στην πρώτη περίπτωση, καλείται η μέθοδος onRestart ενώ στη δεύτερη η onDestroy.

6. **onDestroy**

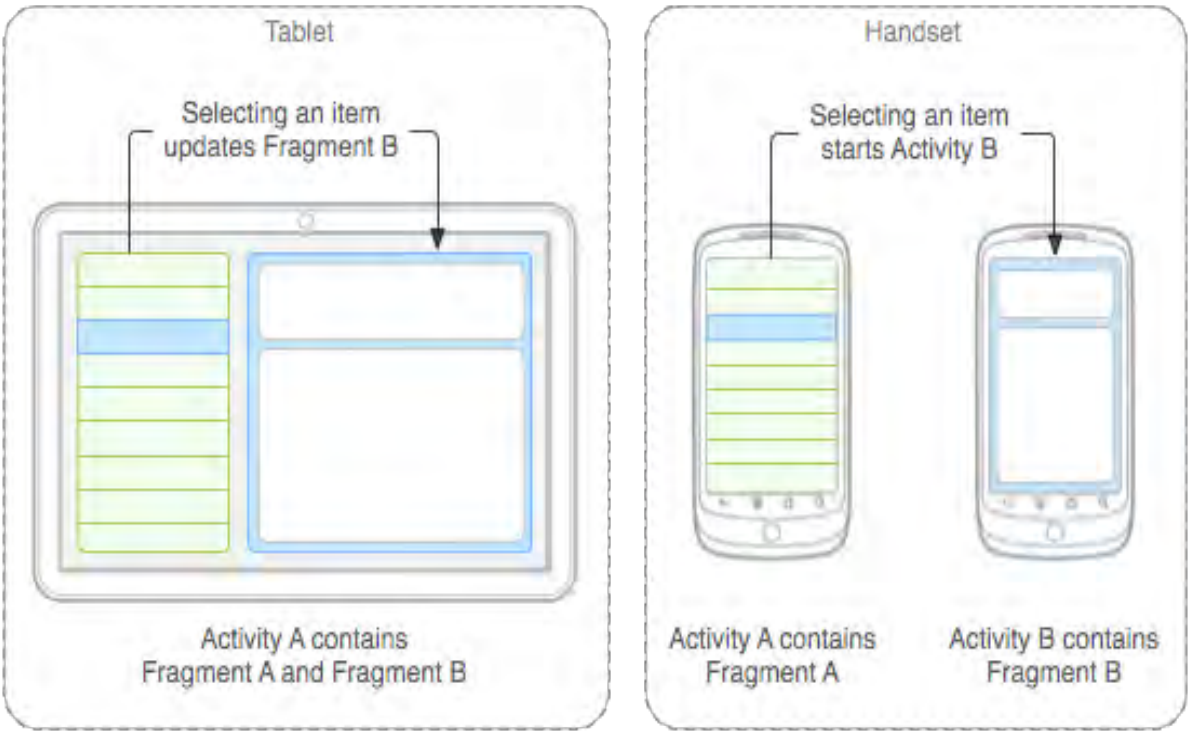
   Η μέθοδος onDestroy καλείται πριν τον τερματισμό του activity, ο οποίος μπορεί να προκλήθηκε είτε επειδή αποδεσμεύεται εντελώς από το χρήστη είτε λόγω αλλαγής διάταξης, όπως η περιστροφή οθόνης, που αναγκάζει το σύστημα να προβεί σε αυτή την ενέργεια. Στο πρώτο σενάριο, η μέθοδος αυτή αποτελεί το τελευταίο callback του κύκλου ζωής του activity ενώ στο δεύτερο δημιουργείται ένα νέο στιγμιότυπο του activity και καλείται και πάλι η onCreate στο καινούργιο αυτό στιγμιότυπο. Το περιεχόμενο της, τέλος, έχει να κάνει με την απελευθέρωση όλων των πόρων που δεν είχαν ήδη αποδεσμευτεί σε προηγούμενα callbacks, όπως η onStop. [7]

Fragments

Τα fragments αποτελούν μέρος ενός activity ενώ συνυπάρχουν ενδεχομένως και με άλλα fragments στην ίδια οθόνη. Ο κύκλος ζωής τους επηρεάζεται άμεσα από αυτόν του activity, έτσι αν το activity τερματιστεί, το ίδιο θα συμβεί και στο fragment.

Τα fragments εισήχθησαν στο Android 3.0 (Api level 11), με σκοπό να προσφέρουν τη δυνατότητα ενός δυναμικού και ευέλικτου UI για μεγαλύτερες οθόνες, πχ. tablets. Κάθε

fragment πρέπει να είναι φτιαγμένο έτσι ώστε να αποτελεί μια αυτόνομη και επαναχρησιμοποιήσιμη οντότητα, με σκοπό να είναι πολύ εύκολο να γίνει η επεξεργασία της εμφάνισης του activity σε εκτελέσιμο χρόνο. [8] Στην Εικόνα 4 φαίνεται πώς ένα σωστά υλοποιημένο fragment συνδυάζεται διαφορετικά με άλλα στοιχεία του activity στο οποίο περικλείεται, ανάλογα με το μέγεθος τη οθόνης.



Εικόνα 4. Παράδειγμα για το πώς δύο ανεξάρτητα modules που περικλείονται σε fragments συνδυάζονται σε ένα activity στην περίπτωση του tablet και σε δύο διαφορετικά activities σε μία κινητή συσκευή [8]

1.2 Η εφαρμογή

Οι ανάγκες που καλύπτει η εφαρμογή Expenses Manager έχουν να κάνουν με τον έλεγχο των χρημάτων του κάθε χρήστη. Είναι πολύ συχνό το φαινόμενο να σπαταλούνται ποσά σε προϊόντα ή υπηρεσίες μέσα στην καθημερινότητα χωρίς να έχει ο αγοραστής την πλήρη εικόνα του συνολικού ποσού που έχει διαθέσιμο, με αποτέλεσμα να έρχεται μια χρονική στιγμή που συνειδητοποιεί ότι διαθέτει πολύ λιγότερα χρήματα από εκείνα που πίστευε και αδυνατεί να εντοπίσει πού τα ξόδεψε, καθώς δεν κατέγραψε τις συναλλαγές του. Με Expenses Manager, του δίνεται η δυνατότητα να εισάγει όλα τα εισοδήματα του και να τα κατηγοριοποιήσει ανάλογα με την πηγή προέλευσης τους. Ταυτόχρονα, ο χρήστης μπορεί να προσθέσει εγγραφές που αντιπροσωπεύουν τα έξοδα του και περιέχουν πληροφορίες όπως την ημερομηνία που έγινε μια συγκεκριμένη αγορά ή το είδος του προϊόντος που αγοράστηκε. Επιπλέον, υπάρχει η επιλογή να δημιουργηθούν προϋπολογισμοί οι οποίοι βασίζονται σε κάποιο συγκεκριμένο έσοδο, έτσι ώστε να καθορίσει ο χρήστης πόσα από αυτά τα χρήματα επιθυμεί να ξοδέψει για ένα συγκεκριμένο σκοπό. Το σύνολο των εσόδων και των εξόδων αθροίζονται έτσι ώστε να εμφανίζεται στην οθόνη το ποσό που τελικά απομένει στον χρήστη.

2 Παρόμοιες Εφαρμογές

Παρακάτω παρουσιάζονται μερικές από τις καλύτερες εφαρμογές που υπάρχουν διαθέσιμες αυτή τη στιγμή στο Google Play Store και χρησιμοποιούνται με σκοπό να οργανώσουν καλύτερα τα έξοδα του χρήστη.

Mint

Πρόκειται για μια από τις παλαιότερες και γνωστότερες εφαρμογές διαχείρισης εξόδων. Με τη σύνδεση των τραπεζικών του λογαριασμών, παρέχονται εργαλεία που αυτόματα καταγράφουν και κατηγοριοποιούν τις συναλλαγές τους χρήστη. Το σπουδαιότερο χαρακτηριστικό της, είναι οι επιλογές που δίνει για τη δημιουργία προϋπολογισμών, όπως οι ειδοποιήσεις σε περίπτωση που κάποιος προϋπολογισμός ξεπεραστεί. Αποτελεί το πιο ολοκληρωμένο εργαλείο για τον έλεγχο την οργάνωση των εξόδων του χρήστη, δυστυχώς όμως η εφαρμογή αυτή δεν είναι διαθέσιμη στις περισσότερες ευρωπαϊκές χώρες, μεταξύ των οποίων και η Ελλάδα. [9]

Wally

Το Wally συνδέεται με τον τραπεζικό λογαριασμό και αντί να ζητάει την εισαγωγή των εξόδων χειροκίνητα από τον χρήστη, τα αποθηκεύει χρησιμοποιώντας φωτογραφίες αποδείξεων και συμπληρώνει αυτόματα κάποιες σχετικές πληροφορίες. Είναι ιδιαίτερα εύκολη στη χρήση και εξοικονομεί χρόνο καθώς δεν απαιτείται η εισαγωγή πολλών στοιχείων. Ωστόσο, ενώ είναι ευρέως διαδεδομένη στις Ηνωμένες Πολιτείες και τον Καναδά, δεν περιλαμβάνει όλες τις τράπεζες και επομένως η λειτουργικότητα της δεν είναι αποτελεσματική στην Ελλάδα. [10]

Spendee

Το Spendee περιέχει πολλά ευκολονόητα γραφήματα και εργαλεία που προσδίδουν μια διορατικότητα για τον τρόπο που ξοδεύονται κάθε μήνα τα χρήματα του χρήστη. Είναι δυνατή η σύνδεση τραπεζικών λογαριασμών και η αυτόματη εισαγωγή όλων των συναλλαγών στην εφαρμογή, ενώ ταυτόχρονα γίνεται να δημιουργηθούν προϋπολογισμοί και λογαριασμοί για πολλαπλούς χρήστες, κάτι το οποίο είναι ιδιαίτερα χρήσιμο για οικογένειες και συγκατοίκους. Εντούτοις, η δωρεάν έκδοση της εφαρμογής περιορίζει τον χρήστη στη δημιουργία ενός μόνο προϋπολογισμού και επιπλέον οι ελληνικές τράπεζες δεν περιλαμβάνονται στη λίστα των τραπεζών που μπορούν να συνδεθούν με την εφαρμογή με αποτέλεσμα όλες οι συναλλαγές να πρέπει να αποθηκεύονται χειροκίνητα. [11]

AndroMoney

Το AndroMoney είναι μία από τις πιο διαδεδομένες και επιτυχημένες εφαρμογές καταγραφής εξόδων στο Google Play Store. Επιτρέπει την μεταφορά χρημάτων σε άλλους λογαριασμούς, την μετατροπή των ποσών σε άλλο συνάλλαγμα και την αποθήκευση των δεδομένων σε Excel ως αντίγραφο ασφαλείας. Επιπλέον περιέχει γραφήματα που αναλύουν τις κινήσεις του χρήστη με λογικό και ευανάγνωστο τρόπο. Τέλος, είναι δωρεάν με εμφάνιση διαφημίσεων, ωστόσο υπάρχει επιλογή αγοράς ετήσιας συνδρομής.

Winbank

Η συγκεκριμένη εφαρμογή ανήκει στην Τράπεζα Πειραιώς και καταγράφει κάθε συναλλαγή που κάνει ο χρήστης με οργανωμένο τρόπο. Επιτρέπει την παρακολούθηση των εξόδων ανά κατηγορία και η ενημέρωση της γίνεται αυτόματα κάθε φορά που ο χρήστης χρησιμοποιεί την κάρτα του. Η λειτουργικότητα της, όμως είναι πολύ περιορισμένη καθώς προσφέρει έλεγχο μόνο για τους λογαριασμούς της Τράπεζας Πειραιώς και δεν επιτρέπει την εισαγωγή άλλων εξόδων ή εσόδων χειροκίνητα.

3 Εργαλεία

Σε αυτό το κεφάλαιο αναφέρονται όλα τα εργαλεία λογισμικού και υλικού που χρησιμοποιήθηκαν για τη δημιουργία της εφαρμογής.

3.1 Android Studio

Το Android Studio είναι το επίσημο ενσωματωμένο προγραμματιστικό περιβάλλον (IDE) για το ομώνυμο λειτουργικό σύστημα της Google. Πρόκειται για τον διάδοχο του Eclipse Android Development Tools (ADT) και είναι χτισμένο πάνω στο λογισμικό του IntelliJ Idea της JetBrains. Παρέχει πολλές λειτουργίες που διευκολύνουν τον προγραμματιστή, όπως γρήγορες διορθώσεις ή εύκολο refactoring, δηλαδή μετονομασία μεταβλητών ή κλάσεων, τροποποίηση των παραμέτρων μια μεθόδου κ.α.

3.2 Java

Οι πιο διαδεδομένες γλώσσες που χρησιμοποιούνται στο προαναφερθέν IDE είναι η Java και η Kotlin. Η δεύτερη μάλιστα, έχει ψηφιστεί ως η προτιμώμενη από τη Google από τον Μάιο του 2019. Αξίζει να αναφερθεί ότι έχουν προκύψει πολλές συζητήσεις σχετικά με το ποια από τις δύο γλώσσες είναι προτιμότερη για την ανάπτυξη Android εφαρμογής. Από τη μία μεριά βρίσκεται η Java, μια από τις πλέον δημοφιλείς γλώσσες με τη διαχείριση των exceptions να είναι από τα δυνατά της σημεία. Από την άλλη, η Kotlin [12] στηρίζεται στη βιβλιοθήκη κλάσεων της Java αλλά χαρακτηρίζεται από τη συνοπτικότητα της, είναι πλήρως διαλειτουργική με τη Java έτσι ώστε να μπορεί να γίνει σταδιακή μετάβαση του υπάρχοντα κώδικα σε Kotlin και επομένως δικαιωματικά θεωρείται ο διάδοχος της Java. Για την ανάπτυξη της συγκεκριμένης εφαρμογής, επιλέχθηκε η Java, καθώς υπάρχει μεγαλύτερη εξοικείωση με αυτή.

3.3 Google Cloud Platform

Το υπολογιστικό νέφος ή cloud computing είναι η παροχή χώρων αποθήκευσης, βάσεων δεδομένων, συστημάτων λογισμικού, υπηρεσιών κ.α. μέσω του Διαδικτύου. Η τεχνολογία αυτή έφερε μεγάλες ευκολίες στους οργανισμούς που υπό άλλες προϋποθέσεις έπρεπε να καταναλώνουν ογκώδη χρηματικά ποσά για την αγορά server και άλλων υπολογιστικών πόρων. Επιπλέον, είναι σημαντικό να ληφθεί υπόψη ότι σε περίπτωση που η εταιρία είχε επιλέξει να αγοράσει τους φυσικούς πόρους, όπως οι servers, θα έπρεπε να ξοδέψει σημαντικό χρόνο και χρήματα στην εγκατάσταση, τη διαμόρφωση και την συντήρηση τους. Ταυτόχρονα, η πλειοψηφία των cloud computing υπηρεσιών παρέχονται με αυτοεξυπηρέτηση και κατ’ απαίτηση (on demand), προσφέροντας με αυτόν τον τρόπο μεγάλη ευελιξία στις

επιχειρήσεις, καθώς χρησιμοποιούν τους πόρους που χρειάζονται για τη δεδομένη στιγμή και όταν υπάρχουν χαμηλότερες απαιτήσεις απελευθερώνουν το υλικό που δεν αξιοποιείται και επομένως χρεώνονται λιγότερο. Τέλος, με το cloud computing η δημιουργία αντιγράφων ασφαλείας γίνεται πιο εύκολη και πιο οικονομική, ενώ η απώλεια δεδομένων είναι σχεδόν αδύνατη καθώς η πληροφορία αντικατοπτρίζεται (mirroring) σε πολλές τοποθεσίες στο δίκτυο. [13] Οι υπηρεσίες που προσφέρει το cloud computing χρεώνονται από τους παρόχους cloud συνήθως ανά χρήση, ωστόσο κάποιες λειτουργίες διατίθενται δωρεάν. Ένας από τους σημαντικότερους παρόχους cloud υπηρεσιών είναι το Google Cloud Platform (GCP).

Το GCP αποτελείται από ένα σύνολο φυσικών στοιχείων, όπως υπολογιστές και μονάδες σκληρού δίσκου, και εικονικούς πόρους, όπως οι εικονικές μηχανές (VM), που περιέχονται στα κέντρα δεδομένων της Google σε όλο τον κόσμο. Κάθε data center βρίσκεται σε μία περιοχή (region), που είναι διαθέσιμη στην Ασία, την Αυστραλία, την Ευρώπη, τη Βόρεια Αμερική και τη Νότια Αμερική, και αποτελείται από ένα σύνολο ζωνών (zones) απομονωμένων μεταξύ τους, οι οποίες έχουν έναν αναγνωριστικό γράμμα.

Οι πόροι GCP που χρησιμοποιούνται από τον χρήστη πρέπει να ανήκουν σε ένα project. Το project αποτελείται από τις ρυθμίσεις, τα δικαιώματα και άλλα metadata που περιγράφουν τις εφαρμογές. Οι πόροι μέσα σε ένα ενιαίο project λειτουργούν μαζί εύκολα ακολουθώντας πάντα τους κανόνες των regions και των zones. [14]

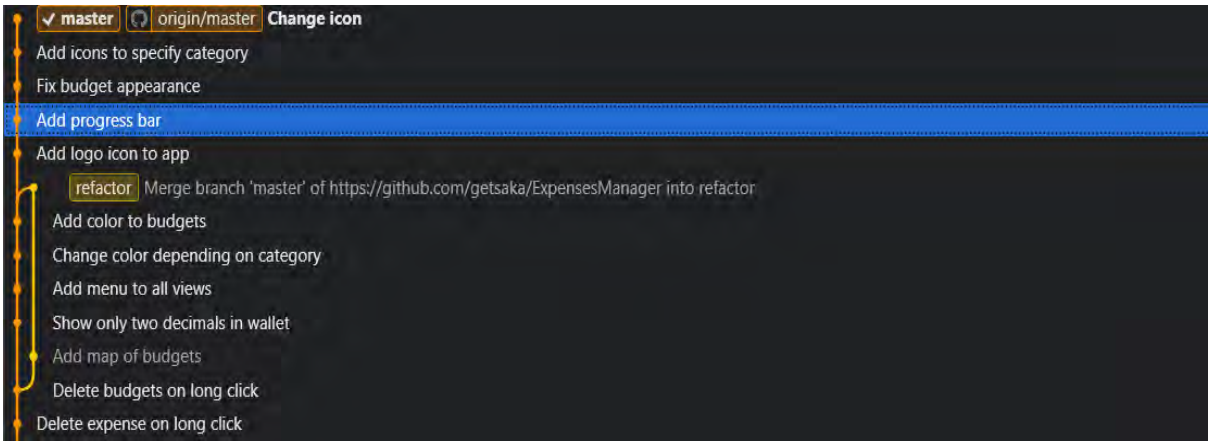
3.4 Fork

Με τον όρο version control εννοούμε ένα λογισμικό που επιτρέπει την διαχείριση αλλαγών με την πάροδο του χρόνου, είτε πρόκειται για κώδικα είτε για άλλου είδους αρχεία. Ο έλεγχος έκδοσης είναι σημαντικός για τους σημερινούς προγραμματιστές, καθώς πέρα από την διαχείριση και την παρακολούθηση των αρχείων, βοηθάει στην ανάπτυξη και ολοκλήρωση του τελικού προϊόντος γρηγορότερα. Ένα από τα συστήματα για version control είναι το git, το οποίο είναι ιδιαίτερα δημοφιλές στους προγραμματιστές και είναι μια καλή επιλογή για μικρές ομάδες, ειδικά εκείνες που εργάζονται πάνω στην ανάπτυξη εφαρμογών ιστού και κινητής τηλεφωνίας. Ένα από τα μεγαλύτερα οφέλη από της σωστής χρήσης του version control είναι το ότι είναι ανιχνεύσιμη οποιαδήποτε αλλαγή ενώ επιτρέπει τη συνεργασία πολλών ατόμων παράλληλα στα ίδια αρχεία χωρίς να υπάρχει κίνδυνος διπλοτύπων ή επεγγραφών (overwriting). [15] Κάποιες από τις σημαντικότερες έννοιες του version control είναι οι παρακάτω:

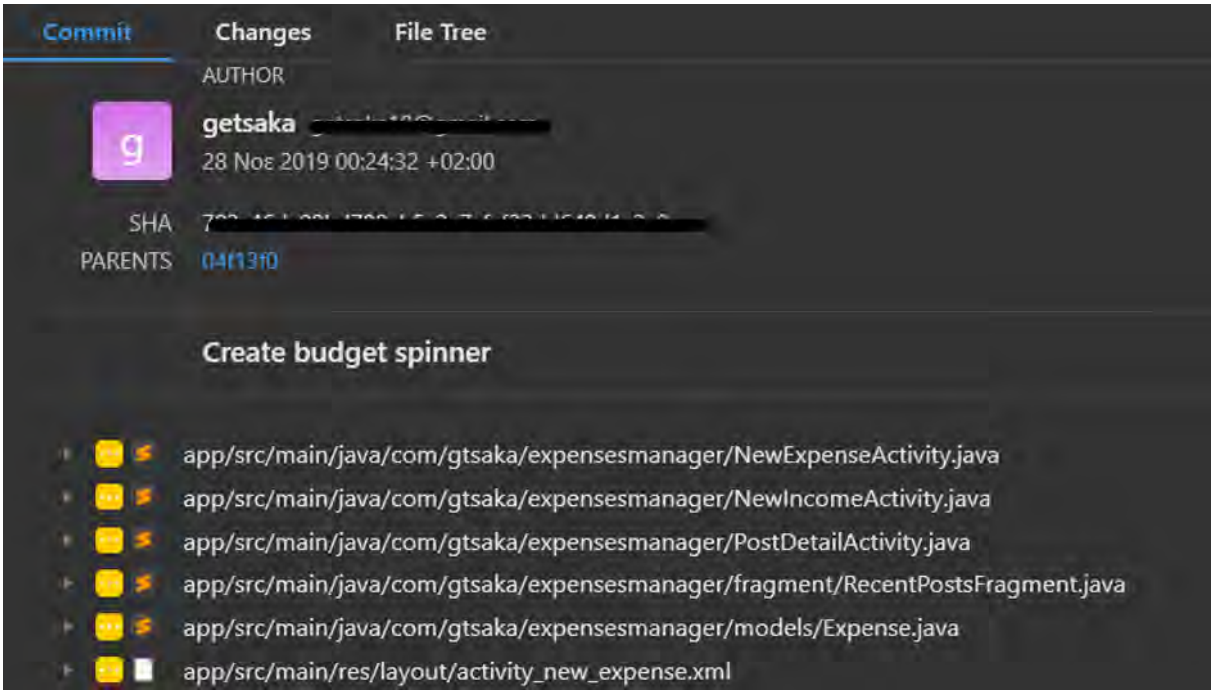
- Branch: Ένα σύνολο αρχείων μπορεί να είναι διακλαδισμένο σε ένα χρονικό σημείο, έτσι ώστε, από εκείνη τη στιγμή και μετά, δύο αντίγραφα αυτών των αρχείων να αναπτύσσονται με διαφορετικές ταχύτητες ή με διαφορετικούς τρόπους ανεξάρτητα το ένα από το άλλο.

- Commit: Ένα commit ισούται με την εγγραφή ή συγχώνευση των αλλαγών που έγιναν στο αντίγραφο του project πάνω στο οποίο δουλεύει ένας προγραμματιστής.
- Conflict: Υπάρχει conflict όταν διαφορετικά μέλη κάνουν αλλαγές στο ίδιο έγγραφο και το σύστημα δεν είναι σε θέση να ενοποιήσει τις αλλαγές. Ένας χρήστης πρέπει να επιλύσει το conflict συνδυάζοντας τις αλλαγές ή επιλέγοντας μία αλλαγή αντί της άλλης.
- Merge: Ως merge ορίζεται η ενέργεια στην οποία δύο ομάδες αλλαγών εφαρμόζονται σε ένα αρχείο ή ένα σύνολο αρχείων.
- Pull, push: Αντιγραφή μιας έκδοσης από ένα repository σε ένα άλλο. Το pull γίνεται για να “τραβήξει” ο χρήστης τις αλλαγές από ένα άλλο branch ή από την κύρια γραμμή προγραμματισμού ενώ αντίστροφα το push στέλνει τις αλλαγές σε άλλο μονοπάτι.
    - Repository: Ο χώρος όπου αποθηκεύονται τα τρέχοντα και τα παλαιότερα δεδομένα των αρχείων, συχνά σε ένα διακομιστή. [16]

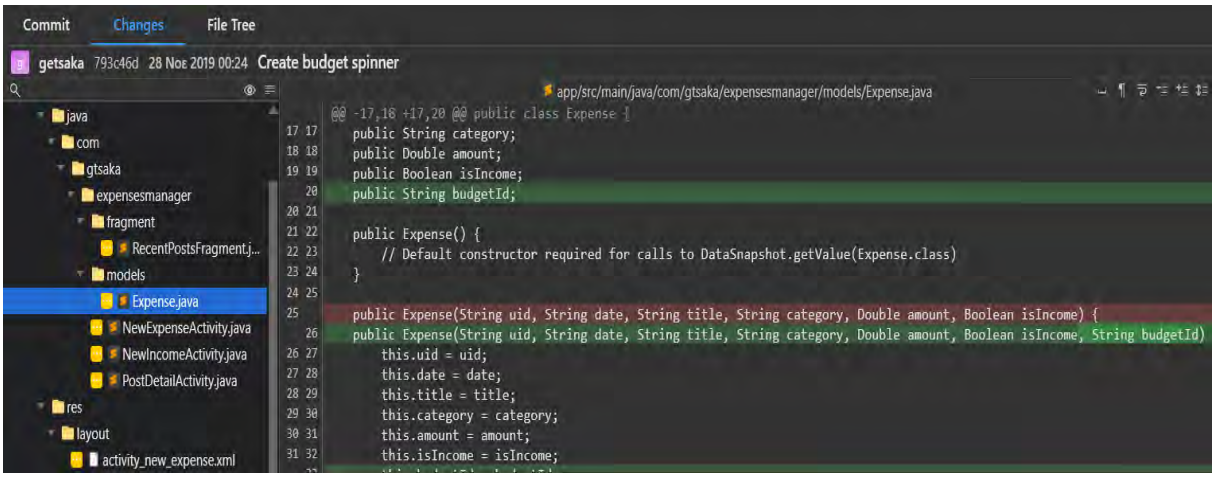
Το πρόγραμμα version control που χρησιμοποιήθηκε για τους σκοπούς του παρόντος project είναι το Fork, το οποίο χρησιμοποιεί το git για να ανιχνεύσει όλες τις αλλαγές που έγιναν. Είναι ιδιαίτερα χρήσιμο καθώς αντικαθιστά τις εντολές του git και επιτρέπει στον χρήστη να κάνει commits, να κάνει επίλυση τυχόν conflicts και πολλά άλλα χρησιμοποιώντας το user interface του Fork. [17]



Εικόνα 5. Παράδειγμα branch στο repository



Εικόνα 6. Παράδειγμα commit



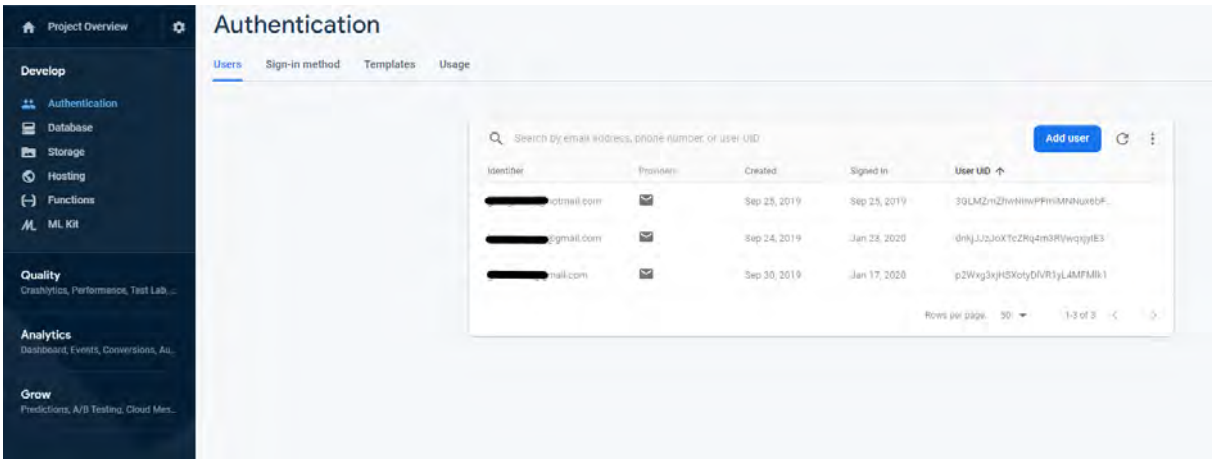
Εικόνα 7. Παράδειγμα εμφάνισης των αλλαγών που έγιναν σε ένα commit

3.5 Firebase

Το Firebase είναι ένα Backend as a service (BaaS) της Google. Με άλλα λόγια, η υπηρεσία αυτή παρέχει εργαλεία που διευκολύνουν τη δημιουργία web και mobile εφαρμογών και τη σύνδεση τους με το cloud χωρίς να χρειάζεται παραμετροποίηση σε servers από τη μεριά του προγραμματιστή. Το Firebase προτιμήθηκε γι' αυτή την εφαρμογή καθώς δεν υπάρχει ανάγκη για επικοινωνία με τρίτους, όπως υπηρεσίες πληρωμής, και αποτελεί την ιδανική λύση για τη δημιουργία μιας ελαφριάς εφαρμογής. Επιπλέον, εφόσον πρόκειται για project του Google Cloud Platform, η επικοινωνία μεταξύ της βάσης και της εφαρμογής καθίσταται πιο αποτελεσματική από την επιλογή μιας υπηρεσίας από άλλο πάροχο cloud. [18]

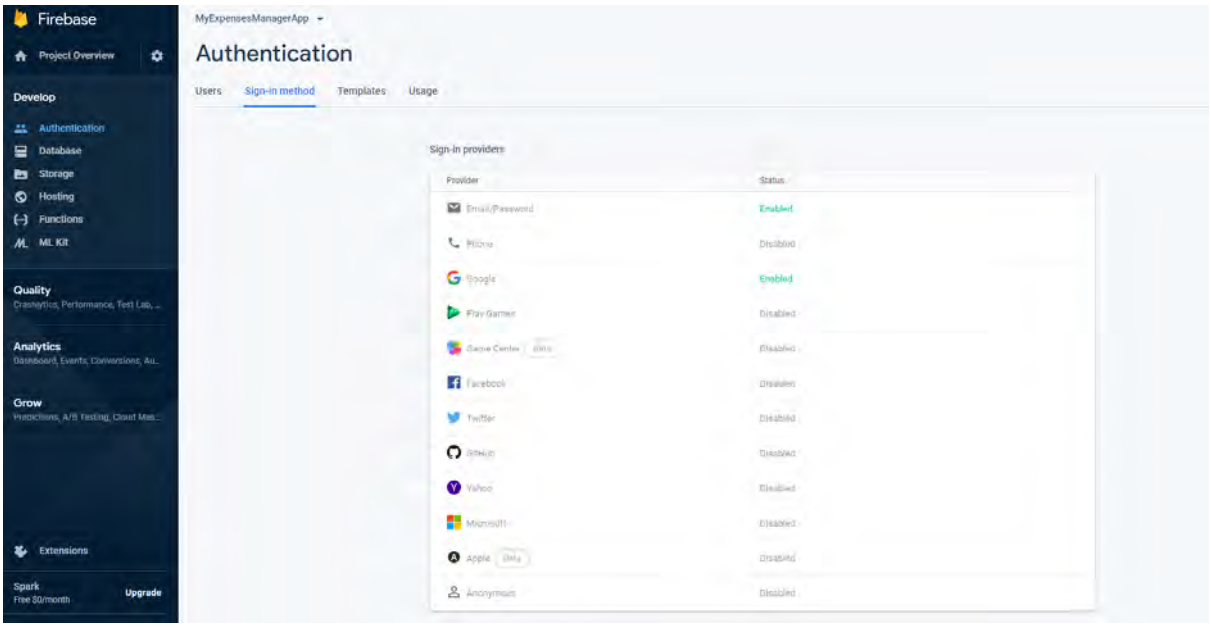
3.5.1 Firebase Authentication

Το Firebase δίνει τη δυνατότητα δημιουργίας λογαριασμών για κάθε χρήστη έτσι ώστε να έχει πρόσβαση μόνο στα δεδομένα που έχει την εξουσιοδότηση. Έτσι την πρώτη φορά που κάποιος θα συνδεθεί στην εφαρμογή θα του ζητηθεί ένα email και ένας κωδικός τα οποία στη συνέχεια θα αποθηκευτούν στο Firebase project και ο χρήστης θα αποκτήσει ένα μοναδικό id. [19]



Εικόνα 8. Οι χρήστες που έχουν δημιουργήσει λογαριασμό στην εφαρμογή

Πέρα από τον συμβατικό τρόπο σύνδεσης μέσω email και κωδικού, υπάρχει η επιλογή να γίνει η σύνδεση ενός χρήστη μέσω άλλης υπηρεσίας, όπως το Facebook ή ο λογαριασμός Google και ο διαχειριστής μπορεί να επιλέξει αν επιτρέπεται ή όχι η σύνδεση με άλλα μέσα.



Εικόνα 9. Τρόποι σύνδεσης με το Firebase

3.5.2 Firebase Real Time Database

Μέσα στις απαιτήσεις της εφαρμογής είναι να υπάρχει δυνατότητα αποθήκευσης των δεδομένων και επομένως είναι απαραίτητο να υπάρχει μία βάση δεδομένων συνδεδεμένη με αυτή. Το Firebase προσφέρει και αυτή την υπηρεσία και πιο συγκεκριμένα δίνει δύο επιλογές βάσεων. Η πρώτη είναι η Real Time Database, η αρχική βάση της Firebase, που παρέχει αποτελεσματικές και γρήγορες λύσεις για εφαρμογές που απαιτούν την ενημέρωση μια αλλαγής σε πραγματικό χρόνο. Το Cloud Firestore από την άλλη, είναι μια πιο καινούργια έκδοση της προαναφερθείσας βάσης και ευνοεί πιο γρήγορες και περιεκτικές αναζητήσεις. Καθώς δεν υπάρχει ανάγκη πολύπλοκης ιεραρχίας, για τις ανάγκες του συγκεκριμένου project προτιμήθηκε η Real Time Database. Τα δεδομένα που αποθηκεύονται ακολουθούν τη μορφοποίηση JSON, όπου ένα αντικείμενο δηλώνεται από άγκιστρα ({}) και στο εσωτερικό του κάθε μεταβλητή ορίζεται με το ζεύγος “name”: “value” χωρισμένες με κόμμα [20].

3.6 Smartphone

Η εφαρμογή προορίζεται για έξυπνα κινητά τηλέφωνα και πιο συγκεκριμένα είναι συμβατή με συσκευές με έκδοση λογισμικού Android 6.0 ή νεότερη. Για το κομμάτι του testing χρησιμοποιήθηκε το μοντέλο OnePlus 6T, με λογισμικό Android 10.0. Παράλληλα χρησιμοποιήθηκαν οι emulators του Android Studio, δηλαδή εικονικές συσκευές που προσομοιώνουν τα χαρακτηριστικά ενός πραγματικού μοντέλου, όπως το Google Pixel 3a με Android 9.0.

3.7 Laptop

Καθώς το Android Studio απαιτεί αρκετά μεγάλη υπολογιστική δύναμη, είναι προτιμώμενη η χρήση του σε υπολογιστές με μεγάλη RAM και ικανοποιητικό επεξεργαστή. Για την παρούσα εργασία χρησιμοποιήθηκε το Lenovo Ideapad Y700 με επεξεργαστή Quad-Core i7 έκτης γενιάς και μνήμη RAM 16 GB DDR4.

4 Η εφαρμογή

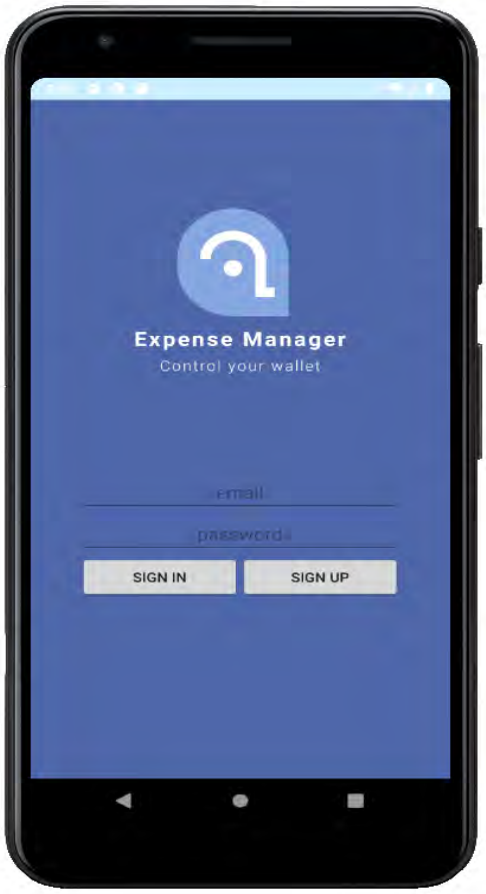
4.1 Εμφάνιση

Η εφαρμογή σχεδιάστηκε χρησιμοποιώντας εργαλεία του Android που χρησιμοποιούνται από τις νεότερες εφαρμογές που υπάρχουν διαθέσιμες έτσι ώστε να έχει σύγχρονη διεπαφή χρήστη.

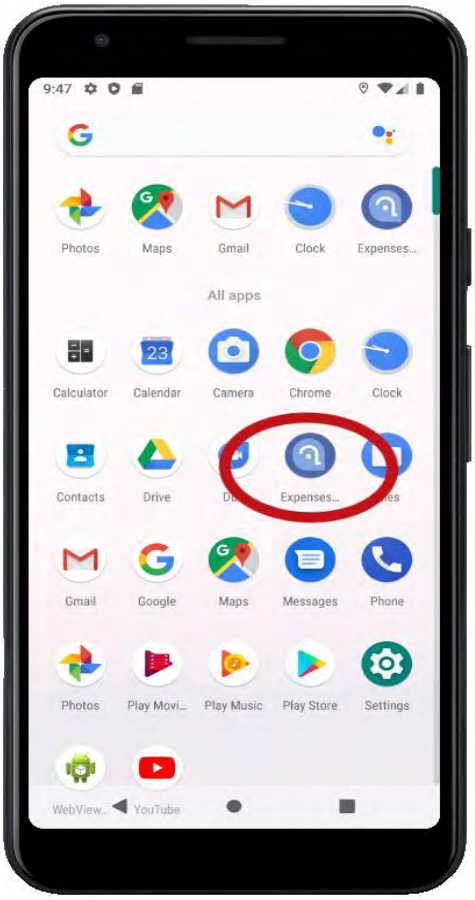
4.1.1 Σύνδεση στην εφαρμογή

Κατά την είσοδο στην εφαρμογή, ο χρήστης βλέπει την οθόνη που τον προτρέπει να δημιουργήσει λογαριασμό ή να συνδεθεί σε κάποιον ήδη υπάρχον. Ζητείται ένα έγκυρο email και ένας κωδικός πρόσβασης.

Χρησιμοποιήθηκε η ιστοσελίδα hatchful [21] για την δημιουργία ενός ελκυστικού λογοτύπου για την εφαρμογή, το οποίο χρησιμοποιήθηκε τόσο για την αρχική οθόνη της εφαρμογής όσο και για το εικονίδιο εκκίνησης.



Εικόνα 10 . Η αρχική οθόνη της εφαρμογής



Εικόνα 11. Το logo της εφαρμογής

Για να εισαχθεί αυτή η εικόνα ως λογότυπο της εφαρμογής ακολουθήθηκαν τα παρακάτω βήματα.

1. Στον φάκελο app/src/main/res πατήστε δεξί κλικ και επιλέξτε **New** > **Image Asset**.
2. Στο πεδίο Icon Type επιλέξτε Launcher Icons (Adaptive and Legacy).
3. Στο πεδίο Layer Name εισάγετε το όνομα που θέλετε να δώσετε στην εικόνα (έστω ic_launcher).
4. Επιλέξτε το πεδίο path και εισάγετε το μονοπάτι που είναι αποθηκευμένη η εικόνα.
5. Πατήστε **Next** και **Finish**.
6. Ανοίξτε το αρχείο AndroidManifest.xml και εισάγετε την παρακάτω σειρά.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.gtsaka.expensesmanager">
   <application
       //...
       android:icon="@mipmap/ic_launcher"
       //...
```

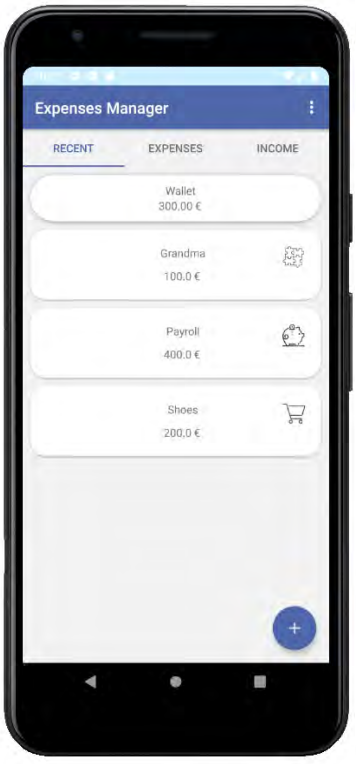
4.1.2 Καρτέλες

Η οθόνη που εμφανίζεται σε έναν συνδεδεμένο χρήστη περιέχει τα πιο πρόσφατα έσοδα και έξοδά του. Για τον ευκολότερο διαχωρισμό τους, δημιουργήθηκαν τρεις καρτέλες.

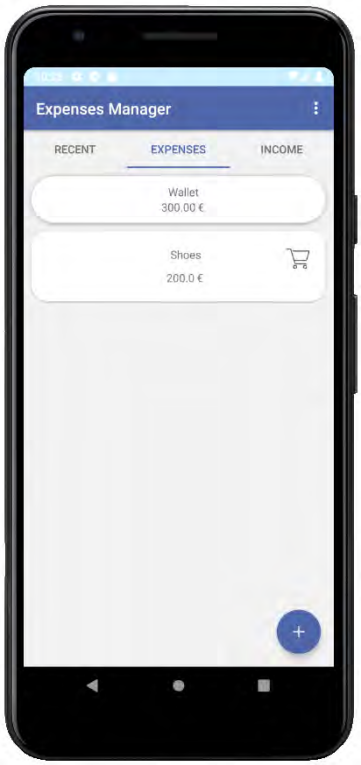
Στην καρτέλα Recent περιέχονται οι εκατό πιο πρόσφατες συναλλαγές ταξινομημένες από την πιο πρόσφατη στην πιο παλιά.

Στην καρτέλα Expenses βρίσκονται όλα τα έξοδα του χρήστη, η κατηγορία των οποίων είναι πιο ευδιάκριτη με την εμφάνιση μιας εικόνας στα δεξιά της κάθε εγγραφής.

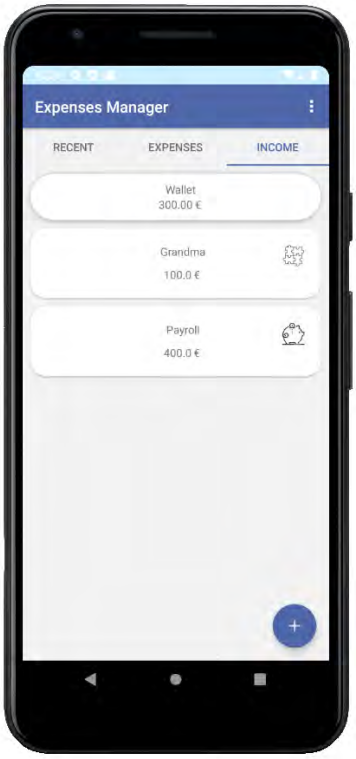
Στην καρτέλα Income εμφανίζονται όλα τα έσοδα του χρήστη, όπως ο μισθός του.



Εικόνα 12. Οι πρόσφατες εγγραφές



Εικόνα 13. Όλα τα έξοδα



Εικόνα 14. Όλα τα έσοδα

Για την εισαγωγή καρτελών στη διάταξη (layout) προσθέστε τον παρακάτω κώδικα στο αντίστοιχο αρχείο xml που περιέχεται στον φάκελο app/src/main/res/layout

```
<com.google.android.material.tabs.TabLayout
   android:id="@+id/tabs"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   app:tabIndicatorColor="@color/colorPrimary"
   app:tabSelectedTextColor="@color/colorPrimary"
   app:tabRippleColor="@color/colorPrimary"/>
```

4.1.3 RecyclerView

Για την εμφάνιση όλων των εσόδων και εξόδων του χρήστη, χρησιμοποιείται το widget RecyclerView. Πρόκειται για ένα container που περιέχει διάφορα components με σκοπό να συνδυαστούν και να εμφανίσουν το επιθυμητό αποτέλεσμα στην οθόνη του χρήστη. Για κάθε εγγραφή δημιουργείται ένα ViewHolder, μια υποκλάση του RecyclerView, και τα references για όλα τα views αποθηκεύονται στη μνήμη. Αυτό σημαίνει ότι δεν είναι αναγκαίο να καλείται η findViewById() κάθε φορά που χρησιμοποιείται η μέθοδος getView() του adapter και επομένως αυξάνεται κατά πολύ η απόδοση της εφαρμογής. Τα view holder objects τα διαχειρίζεται ο recycler view adapter. ο οποίος τα δημιουργεί και τα συνδέει με τα δεδομένα που πρέπει να προβληθούν στο χρήστη ανάλογα με τις απαιτήσεις του. Επιπλέον, επιλέχθηκε το LinearLayoutManager για την εμφάνιση των εγγραφών σε κυλιόμενη λίστα με κάθετο προσανατολισμό. [22]

Μια άλλη δημοφιλής επιλογή για την εμφάνιση λίστας είναι το widget ListView, ωστόσο χρησιμοποιούνταν ως επί το πλείστον πριν την εμφάνιση του RecyclerVIew στην έκδοση του Android 5.0 (Lollipop). [23] Ενώ, επομένως, είναι πιο εύκολο και απλό στη χρήση, είναι σημαντικά πιο αργό και λιγότερο ευέλικτο από το RecyclerView.

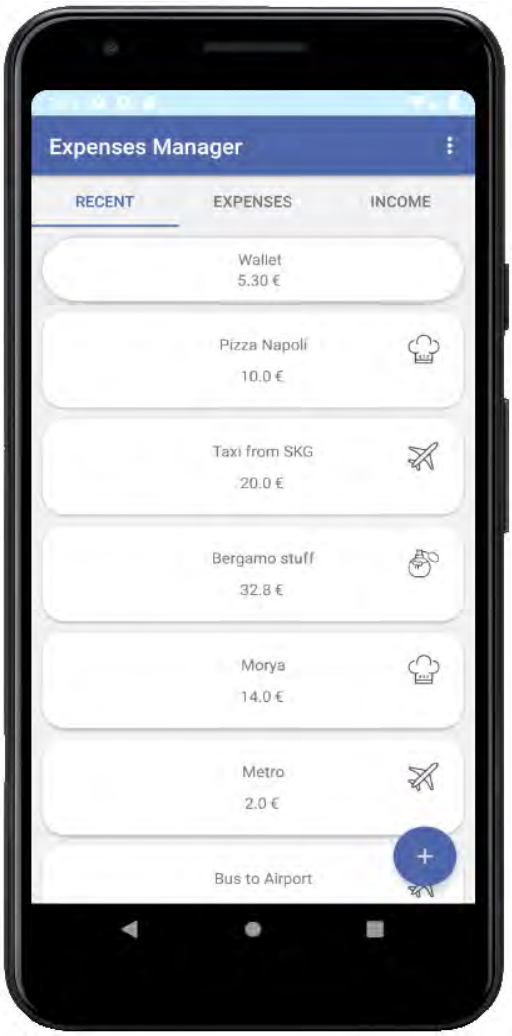
Για την εισαγωγή του RecyclerView στο layout που εμφανίζει όλους τους προϋπολογισμούς προστέθηκε ο παρακάτω κώδικας.

```
<androidx.recyclerview.widget.RecyclerView
   android:id="@+id/budgetList"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
```

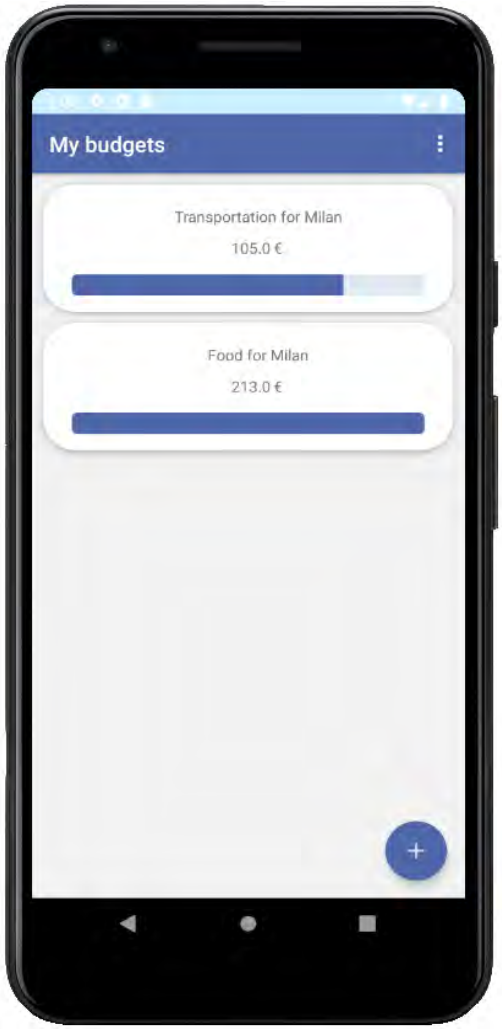
```
android:layout_alignParentTop="true"
android:clipToPadding="false"
android:padding="5dp"
android:scrollbars="vertical"
tools:listitem="@layout/content_budgets" />
```

4.1.4 CardView

Κάθε σειρά του RecyclerView που εμφανίζεται, το οποίο ορίζεται με τη σειρά tools:listitem στο RecyclerView widget, περιέχεται μέσα σε ένα card view, ένα container που προσφέρει οπτικές βελτιστοποιήσεις συγκριτικά με τα layouts, όπως στρογγυλεμένες γωνίες και σκίαση. Το αποτέλεσμα είναι μια συμπαγής, ανεξάρτητη και απομονωμένη κάρτα για κάθε εγγραφή. [24] Κάθε CardView περιέχει τον τίτλο του εσόδου ή εξόδου, το ποσό που του αναλογεί και μία εικόνα η οποία υποδεικνύει την κατηγορία της εγγραφής.



Εικόνα 15. RecyclerView που εμφανίζει CardView



Εικόνα 16. CardViews για την εμφάνιση των προϋπολογισμών

Για την υλοποίηση αυτού του widget γράφτηκε ο παρακάτω κώδικας xml.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_margin="5dp"
    android:id="@+id/post_id"
    app:cardCornerRadius="20dp">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp">
        <ImageView
            android:id="@+id/category_image"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_alignParentTop="true"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:padding="10dp"
            android:src="@drawable/transport" />
        <LinearLayout
            android:id="@+id/starLayout"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/postAuthorLayout"
            android:layout_alignParentRight="true"
            android:layout_alignTop="@+id/postAuthorLayout"
            android:gravity="center_vertical"
            android:orientation="horizontal">
        </LinearLayout>
        <include layout="@layout/include_post_text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_below="@+id/postAuthorLayout"
            android:layout_marginLeft="5dp"
            android:layout_marginTop="10dp" />
    </RelativeLayout>
</androidx.cardview.widget.CardView>
```

Αντίστοιχα για τους προϋπολογισμούς έχει χρησιμοποιηθεί το ίδιο widget και σε αυτό περιλαμβάνεται ο τίτλος και το ποσό του προϋπολογισμού καθώς και μια μπάρα που δείχνει ποσοστιαία πόσο από το αρχικό ποσό του προϋπολογισμού έχει δαπανηθεί.

4.1.5 Πορτοφόλι

Ένα άλλο στοιχείο που εμφανίζεται στην εφαρμογή είναι το πορτοφόλι, δηλαδή το συνολικό ποσό που απομένει στον χρήστη με βάση τα έσοδα και έξοδα που έχει εισάγει. Το πορτοφόλι είναι και αυτό ένα cardView widget το οποίο εισάγουμε στο layout που περιέχει τις καρτέλες και όλες τις εγγραφές του χρήστη, με τον παρακάτω κώδικα.

```
<include
   android:id="@+id/wallet_card_view"
   layout="@layout/include_wallet"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_alignParentLeft="true"
   android:layout_below="@id/tabs"
   android:layout_marginLeft="10dp"
   android:layout_marginRight="10dp"
   android:layout_marginTop="10dp" />
```

4.1.6 Floating Button

Στην κάτω δεξιά γωνία υπάρχει το floating button (FAB), ένα κυκλικό κουμπί που ενεργοποιεί την κύρια ενέργεια στο περιβάλλον χρήστη της εφαρμογής, δηλαδή την δημιουργία εσόδων και εξόδων. Το floating button εισάγεται στο layout με τον παρακάτω κώδικα.

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
   android:id="@+id/fabNewPost"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentBottom="true"
   android:layout_alignParentRight="true"
   android:src="@drawable/ic_new_button"
   android:layout_margin="16dp"/>
```

4.2 Setup

4.2.1 Δημιουργία νέου android project

Αρχικά είναι απαραίτητη προϋπόθεση να δημιουργηθεί ένα νέο project στο Android Studio.

1. Στην οθόνη εκκίνησης του Android Studio επιλέξτε **Start a new Android Studio project**.
2. Στο παράθυρο **Choose your project**, επιλέξτε **Empty Activity** και πατήστε **Next**.
3. Στο παράθυρο **Configure your project** συμπληρώστε το όνομα της εφαρμογής, το όνομα του πακέτου της, το οποίο πρέπει να είναι μοναδικό, καθώς και τον φάκελο όπου θέλετε να αποθηκεύσετε το project. Στο Language μενού επιλέξτε τη γλώσσα **Java** και τσεκάρετε το κουτί δίπλα στο **Use androidx.* artifacts**.
4. Πατήστε **Finish**.

Μετά από λίγο χρόνο επεξεργασίας, θα ανοίξει το κύριο παράθυρο του Android Studio. [25]

4.2.2 Δημιουργία upload key και keystore

Πριν την εγκατάσταση ή ενημέρωση μια εφαρμογής, είναι απαραίτητη η ύπαρξη ψηφιακής υπογραφής έτσι ώστε να είναι δυνατή η μεταφόρτωση της στο Google Play. Παρακάτω ορίζονται κάποιες έννοιες έτσι ώστε να γίνει πιο κατανοητή η διαδικασία. [26]

Πιστοποιητικά

Ένα πιστοποιητικό δημοσίου κλειδιού (public key certificate) περιέχει το δημόσιο κλειδί (public key) ενός ζευγαριού δημόσιου / ιδιωτικού κλειδιού (αρχεία .der ή .pem), καθώς και ορισμένα άλλα μεταδεδομένα που προσδιορίζουν τον κάτοχο (για παράδειγμα όνομα και τοποθεσία) που κατέχει το αντίστοιχο ιδιωτικό κλειδί (private key). Κατά την υπογραφή της εφαρμογής, το πιστοποιητικό επισυνάπτεται σε αυτή και συσχετίζει την εφαρμογή με τον κάτοχο και το αντίστοιχο ιδιωτικό του κλειδί. Αυτό βοηθά το Android να διασφαλίσει ότι τυχόν μελλοντικές ενημερώσεις της εφαρμογής είναι αυθεντικές και προέρχονται από τον αρχικό συντάκτη. Το κλειδί που χρησιμοποιείται για τη δημιουργία αυτού του πιστοποιητικού ονομάζεται κλειδί υπογραφής εφαρμογής (app signing key). Κάθε εφαρμογή πρέπει να χρησιμοποιεί το ίδιο πιστοποιητικό καθ’ όλη τη διάρκεια της ζωής της, προκειμένου οι χρήστες να μπορούν να εγκαταστήσουν νέες εκδόσεις της.

Δακτυλικό αποτύπωμα πιστοποιητικού

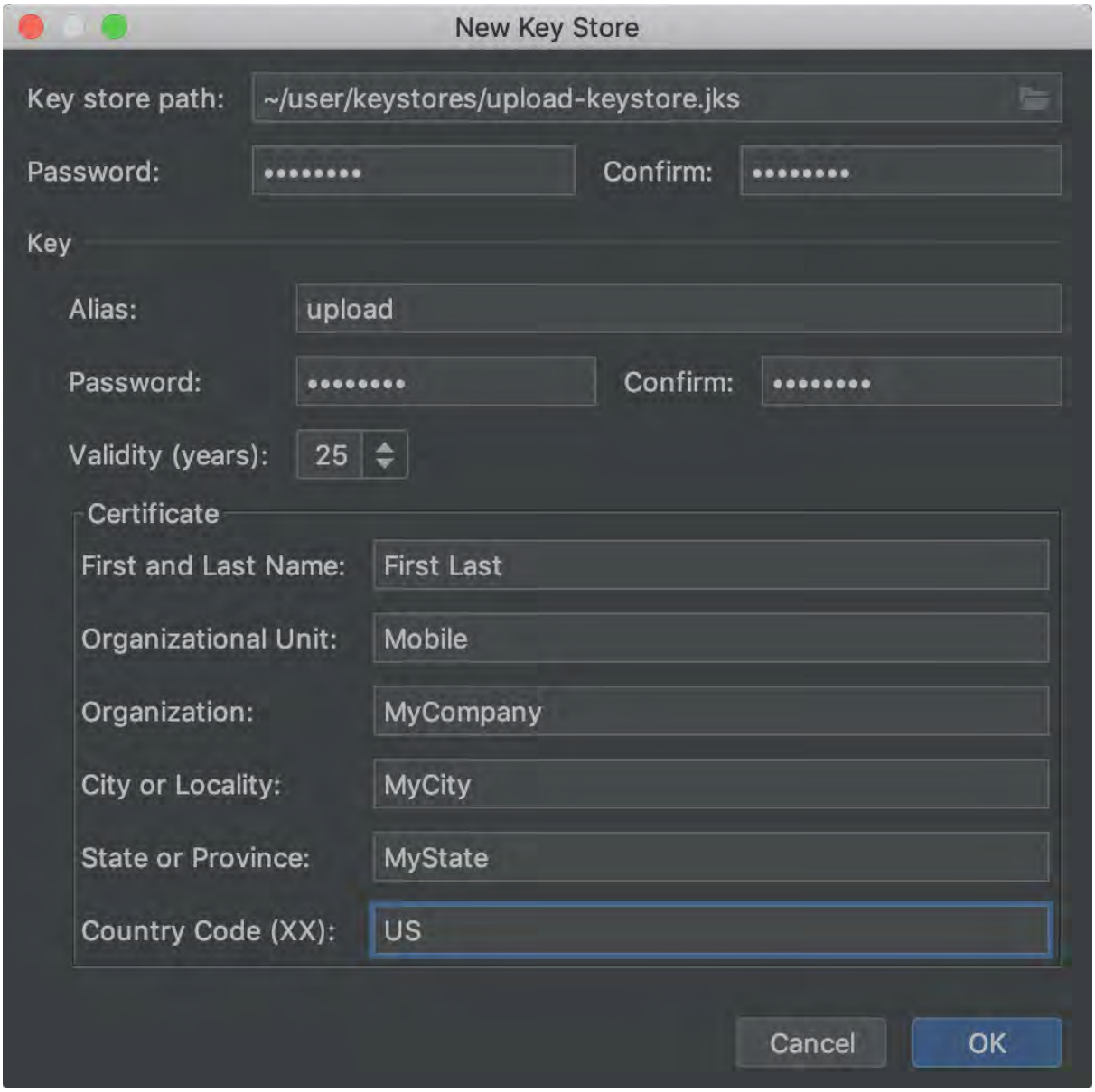
Ένα δακτυλικό αποτύπωμα πιστοποιητικού (certificate fingerprint) είναι μια σύντομη και μοναδική αναπαράσταση ενός πιστοποιητικού που συχνά ζητείται από τους παρόχους API μαζι με το όνομα του πακέτου (package name) για να καταχωρήσει μια εφαρμογή η οποία θέλει να χρησιμοποιήσει τις υπηρεσίες του.

Κλειδιά

App signing key: Το κλειδί που χρησιμοποιείται για την υπογραφή των APK που είναι εγκατεστημένα στη συσκευή ενός χρήστη. Για την ασφαλή ενημέρωση της εφαρμογής, το κλειδί αυτό δεν αλλάζει ποτέ κατά τη διάρκεια της ζωής της. Το app signing key είναι ιδιωτικό και πρέπει να παραμείνει μυστικό, ωστόσο είναι δυνατό να κοινοποιηθεί το πιστοποιητικό που δημιουργείται χρησιμοποιώντας αυτό το κλειδί.

Upload key: Το κλειδί που χρησιμοποιείται για την υπογραφή που χρειάζεται πριν τη μεταφόρτωση της εφαρμογής στο Google Play. Πρέπει να κρατήσετε το κλειδί μεταφόρτωσης μυστικό. Πρόκειται και πάλι για ένα ιδιωτικό κλειδί που πρέπει να παραμείνει μυστικό, ωστόσο είναι δυνατό να κοινοποιηθεί το πιστοποιητικό που δημιουργείται χρησιμοποιώντας το.
Με σκοπό να διατηρηθούν ασφαλή τα κλειδιά της εφαρμογής, συνιστάται το app signing key και το upload key να είναι διαφορετικά.
Για τη δημιουργία του upload key, ακολουθήθηκαν τα παρακάτω βήματα

1. Από το μενού, επιλέξτε **Build > Build > Generate Signed Bundle/APK**.
2. Στο παράθυρο **Generate Signed Bundle or APK**, επιλέξτε **Android App Bundle** ή **APK** και πατήστε **Next**.
3. Κάτω από το πεδίο **Key store path**, πατήστε **Create new**.
4. Στο παράθυρο **New Key Store**, εισάγετε τις πληροφορίες που ζητούνται όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 17. Δημιουργία νέου upload key και keystore στο Android Studio [26]

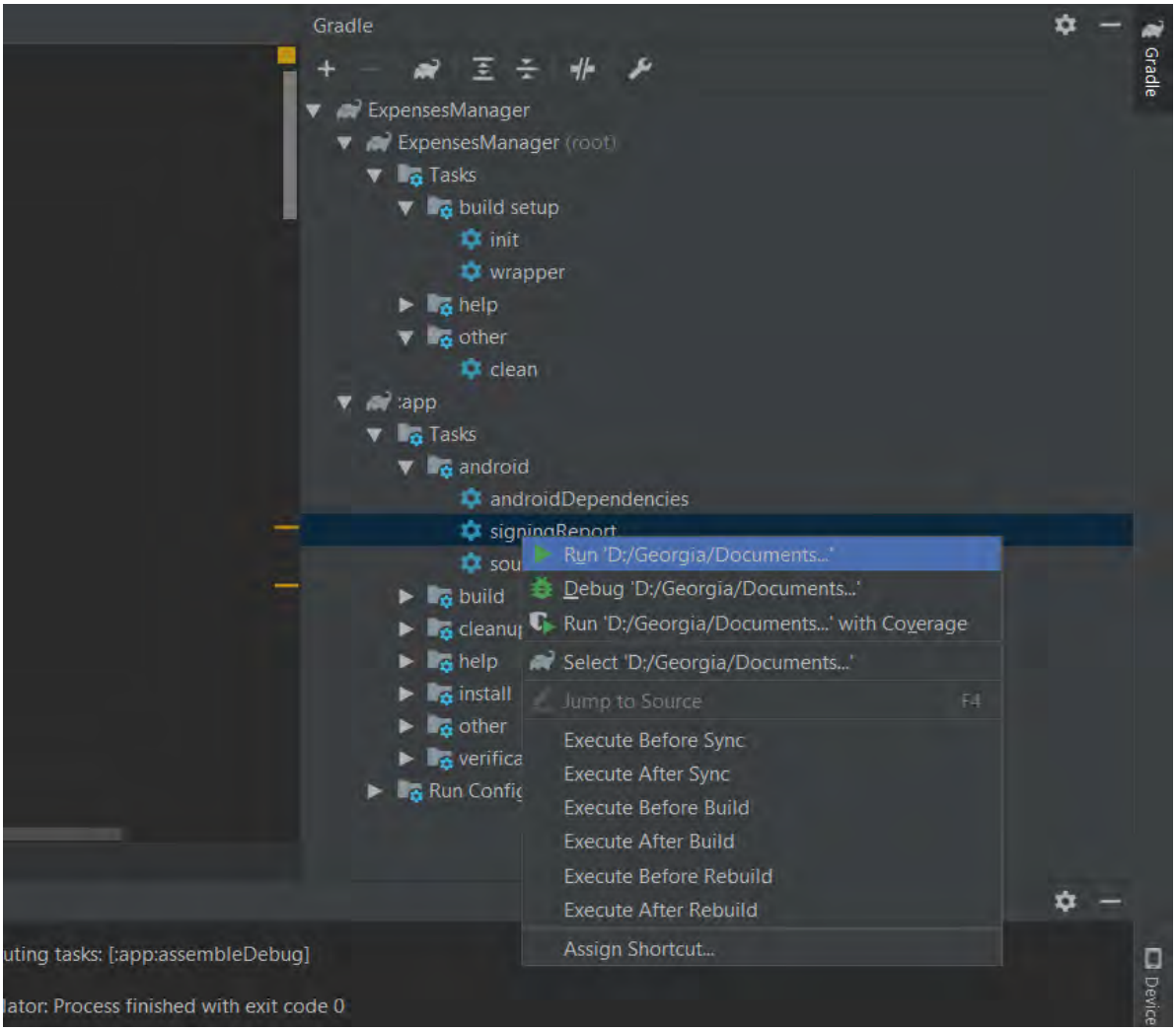
5. **Key store path:** Επιλέξτε την τοποθεσία που θέλετε να αποθηκευτεί το keystore.
6. **Password:** Δημιουργήστε έναν ασφαλή κωδικό για το keystore.
7. **Alias:** Θέστε ένα αναγνωριστικό όνομα για το κλειδί σας
8. **Password:** Δημιουργήστε έναν ασφαλή κωδικό για το κλειδί, διαφορετικό από αυτόν που χρησιμοποιήθηκε για το keystore.
9. **Validity (years):** Θέστε το χρονικό διάστημα που επιθυμείτε να είναι έγκυρο το κλειδί σας (τουλάχιστον 25 χρόνια έτσι ώστε να καλυφθεί η διάρκεια ζωής της εφαρμογής).
10. **Certificate:** Εισάγετε κάποιες προσωπικές πληροφορίες για το πιστοποιητικό. Αυτά τα στοιχεία δεν εμφανίζονται στην εφαρμογή αλλά περιέχονται στο πιστοποιητικό.
11. Πατήστε **OK**.

12. Πατήστε **Cancel** για να παράγετε το κλειδί και το keystore χωρίς να συνεχίσετε στην υπογραφή της εφαρμογής.

Εύρεση του SHA-1 Signing-certificate fingerprint

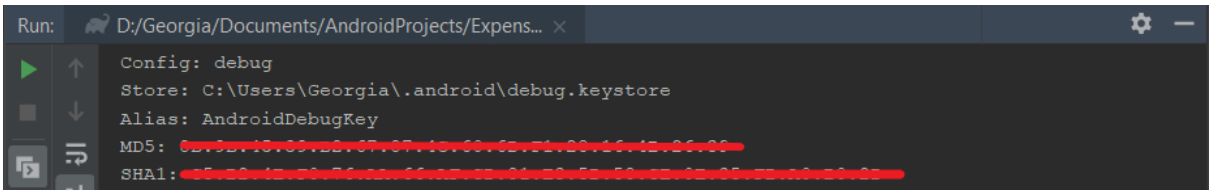
Με τα βήματα που ακολουθήθηκαν παραπάνω, προέκυψε ένα μοναδικό δακτυλικό αποτύπωμα που αντιστοιχεί σε αυτή την υπογραφή και είναι κρυπτογραφημένο με την κρυπτογραφική συνάρτηση κατακερματισμού SHA-1. Η συνάρτηση αυτή μετατρέπει την είσοδο σε μια κατακερματισμένη τιμή μεγέθους 160 bit γνωστό και ως 'σύνοψη' (message digest). [27]

Η τιμή της σύνοψης ζητείται κατά τη δημιουργία νέου project στο Google Cloud Platform (GCP) με σκοπό να παραχθεί ένας OAuth2 client και ένα API key για την εφαρμογή. Το OAuth2 είναι ένα πλαίσιο εξουσιοδότησης που επιτρέπει σε εφαρμογές να έχουν περιορισμένη πρόσβαση σε λογαριασμούς χρηστών. [28] Για την εύρεση, επομένως του κρυπτογραφημένου δακτυλικού αποτυπώματος, εκτελείται το αρχείο signingReport που βρίσκεται στο Graddle menu του Android Studio, όπως φαίνεται στην Εικόνα 18.



Εικόνα 18. Path του αρχείου signingReport

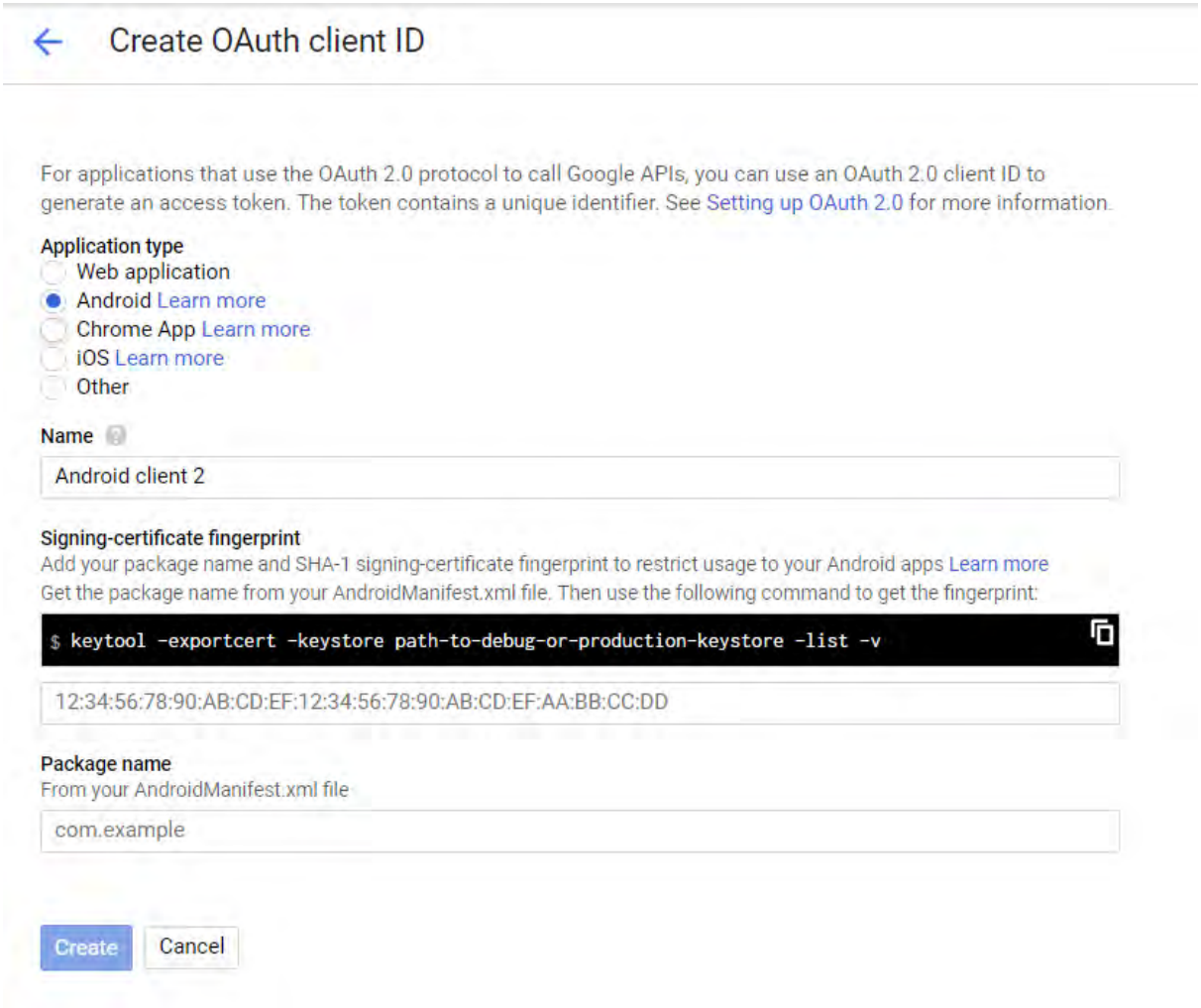
Το SHA-1 key εμφανίζεται στην κονσόλα στο κάτω μέρος του Android Studio. [29]



Εικόνα 19. Εύρεση του κλειδιού SHA-1

4.2.3 Δημιουργία νέου project στο Google Cloud Platform

Μετά τη δημιουργία λογαριασμού στο Google Cloud Platform (GCP), επιλέγεται η δημιουργία νέου project και στη συνέχεια από το μενού ακολουθούνται τα βήματα: Credentials -> Create Credentials -> OAuth client ID για Android. Εκεί ζητείται το κλειδί SHA-1 key που βρέθηκε παραπάνω και το package name που βρίσκεται στο AndroidManifest.xml αρχείο του project.

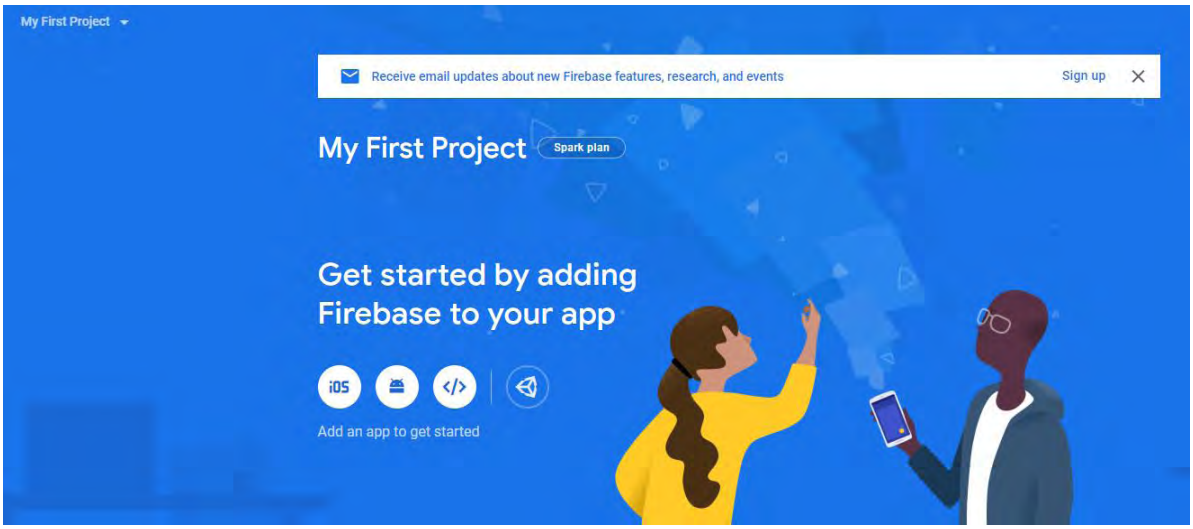


Εικόνα 20. Σύνδεση GCP και Android

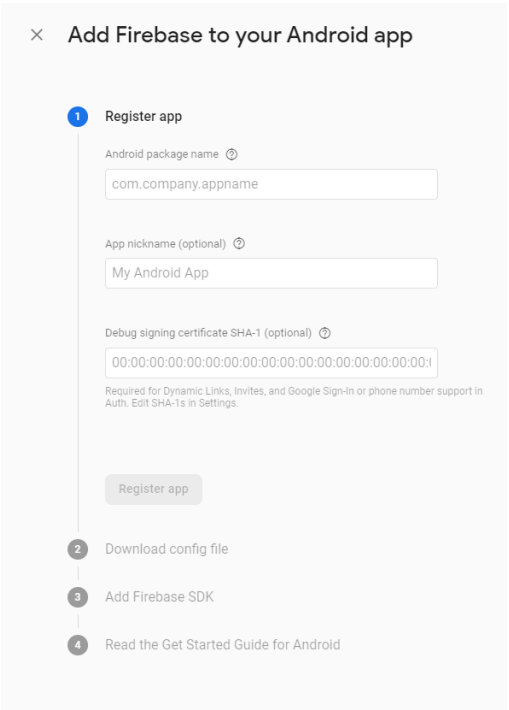
4.2.4 Σύνδεση με το Firebase

Για τη σύνδεση του Firebase με το android project ακολουθήθηκαν τα παρακάτω βήματα:

1. Δημιουργία Firebase project: Από την κονσόλα του Firebase, επιλέξτε **Add Project** και επιλέξτε ένα όνομα για το καινούργιο project.
2. Καταχώριση της εφαρμογής στο Firebase: Στο κέντρο της σελίδας επισκόπησης του project, κάντε κλικ στο εικονίδιο Android.



Εικόνα 21. Αρχική οθόνη του Firebase project



Εικόνα 22. Βήματα σύνδεσης του Firebase με την εφαρμογή Android

Εισάγετε το όνομα του πακέτου της εφαρμογής σας στο πεδίο Android package name και το SHA-1 key που βρέθηκε παραπάνω. Επιλέξτε το κουμπί **Register app**.

3. Προσθέστε το Firebase configuration file στην εφαρμογή: Στην επιλογή **Download config file** πατήστε το κουμπί Download google-services.json, το οποίο μπορείτε να ξανακατεβάσετε οποιαδήποτε στιγμή. Μετακινήστε το αρχείο ρυθμίσεων στον φάκελο app του project. Για να ενεργοποιήσετε τα προϊόντα Firebase στην εφαρμογή, προσθέστε το google-services plugin στα Gradle αρχεία. Πιο συγκεκριμένα, στο project-level Gradle αρχείο (build.gradle) προσθέστε ή βεβαιωθείτε ότι υπάρχουν τα παρακάτω.

```
buildscript {

  repositories {
    google()  // Google's Maven repository
  }

  dependencies {
    // ...
    // Google Services plugin
    classpath 'com.google.gms:google-services:4.3.3'
  }
}

allprojects {
  // ...

  repositories {
    google()  // Google's Maven repository
    // ...
  }
}
```

Στο app-level Gradle αρχείο (app/build.gradle) προσθέστε τα παρακάτω.

```
apply plugin: 'com.android.application'
// Add the following line:
apply plugin: 'com.google.gms.google-services'  // Google Services plugin

android {
  // ...
}
```

4. Προσθέστε τα Firebase SDKs στην εφαρμογή: Είναι δυνατό να προσθέσετε οποιοδήποτε από τα υποστηριζόμενα προϊόντα του Firebase σε μια Android εφαρμογή προσθέτοντας στο αρχείο app/build.gradle τα dependencies που σχετίζονται με αυτά τα προϊόντα.

```
dependencies {
  // ...

  // Add the Firebase SDK for Google Analytics
  implementation 'com.google.firebase:firebase-analytics:17.2.2'

  // Add the SDKs for other Firebase products you want to use in your app
  implementation 'com.google.firebase:firebase-auth:19.2.0'
  implementation 'com.google.firebase:firebase-firestore:21.3.1'
  implementation 'com.google.firebase:firebase-database:19.2.0'
}
```

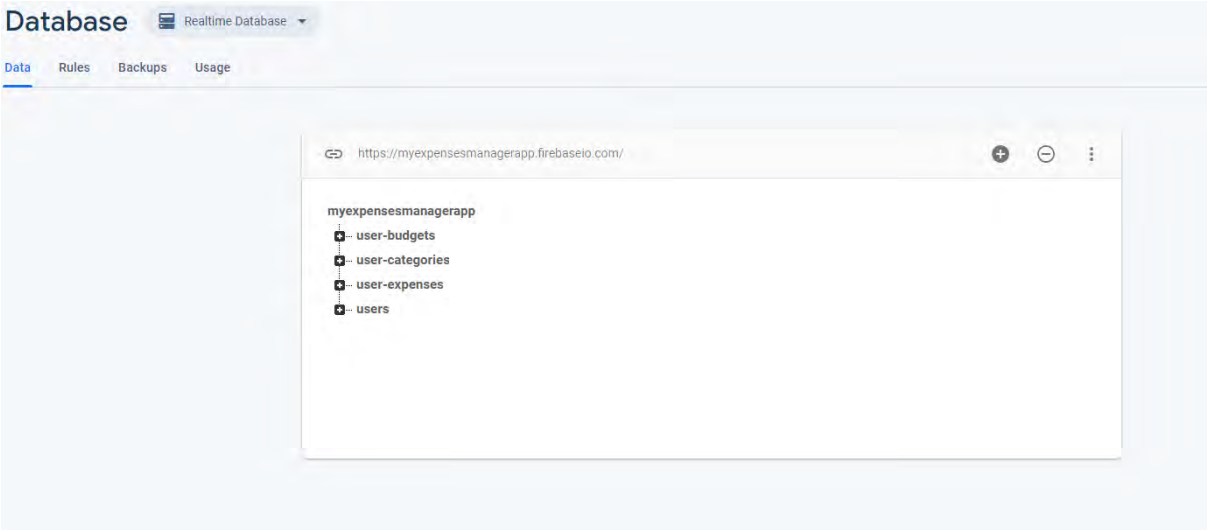
Πλέον υπάρχει επικοινωνία μεταξύ της εφαρμογής και του Firebase. [30]

Με στόχο να διασφαλιστεί η πρόσβαση κάθε χρήστη μόνο στις εγγραφές που έχει δημιουργήσει ο ίδιος, δημιουργήθηκε ο παρακάτω κανόνας εγγραφής δεδομένων, γραμμένος σε JSON.

```
{
  "rules": {
    // User profiles are only readable/writable by the user who owns it
    "users": {
      "$UID": {
        ".read": "auth.uid == $UID",
        ".write": "auth.uid == $UID"
      }
    },
      "user-categories": {
      ".read": true,
      "$UID": {
        "$CATID": {
        ".write": "auth.uid == $UID",
        }
      }
    },
    // User posts can be read by anyone but only written by the user that owns it,
    // and with a matching UID
    "user-expenses": {
      ".read": true,
      "$UID": {
        "$POSTID": {
          ".write": "auth.uid == $UID",
            ".validate": "data.exists() || newData.child('uid').val() == auth.uid",
```

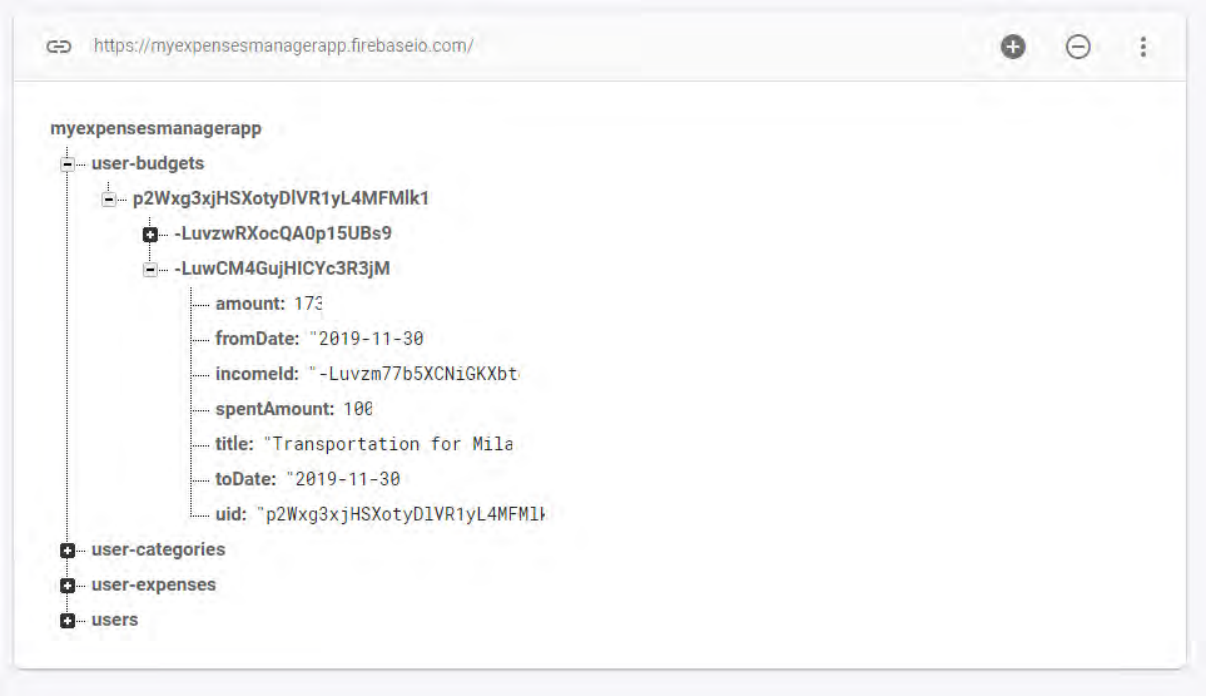
```
      },
      ".indexOn": ["isIncome_date","date"]
    }
  },
  "user-budgets": {
    ".read": true,
    "$UID": {
      "$BUDGETID": {
      ".write": "auth.uid == $UID",
      ".validate": "data.exists() || newData.child('uid').val() == auth.uid",

      }
    }
  },

 }
}
```

Τελικά στη βάση αποθηκεύονται μόνο πληροφορίες που ακολουθούν το παραπάνω μοτίβο. Στην κατηγορία users υπάρχουν όλοι οι εγγεγραμμένοι χρήστες και το μοναδικό αναγνωριστικό τους (uid). Αντίστοιχα, στις κατηγορίες user-expenses, user-categories και user-budgets υπάρχουν οι εγγραφές για έξοδα/έσοδα, κατηγορίες και προϋπολογισμούς ανά uid. Παρακάτω φαίνεται η οργάνωση των δεδομένων όπως καθορίζεται από τον κανόνα.



Εικόνα 23. Η δομή της βάσης δεδομένων

Για παράδειγμα, έστω ότι ένας χρήστης δημιουργεί ένα καινούργιο προϋπολογισμό. Αυτός θα αποθηκευτεί κάτω από την κατηγορία user-budgets, μέσα στο μονοπάτι που αντιστοιχεί σε αυτόν.



Εικόνα 24. Παράδειγμα εγγραφής στη βάση δεδομένων

4.2.5 Version Control

Όπως αναφέρθηκε παραπάνω, κατά τη διάρκεια ανάπτυξης της εφαρμογής χρησιμοποιήθηκε το Fork ως εργαλείο version control, το οποίο χρησιμοποιεί το git. Γι' αυτό το λόγο χρειάζεται ένας λογαριασμός στο Github με σκοπό να ανέβουν οι αλλαγές και στο github repository. Μετά τη δημιουργία νέου λογαριασμού (ή τη χρήση υπάρχοντος), τα βήματα που ακολουθήθηκαν είναι τα εξής:

1. Μέσα στον φάκελο του Android project εκτελέστε τις εντολές
    a. git init
    b. git add .
2. Από το πρόγραμμα Fork που εγκαταστάθηκε παραπάνω, ανοίξτε το Android project το οποίο πλέον υποστηρίζει το version control. Κάντε το πρώτο commit με όλα τα αρχεία που υπάρχουν μέσα σε αυτό.
3. Εκτελέστε τις εντολές
    a. git remote add origin https://github.com/getsaka/ExpensesManager.git
    b. git push -u origin master

Πλέον κάθε αλλαγή είναι ανιχνεύσιμη από το Fork και το repository στο Github ενημερώνεται κάθε φορά που γίνεται push.

4.2.6 Code Style

Ο κώδικας που γράφτηκε υπακούει στους κανόνες που ορίζει η Google. [31] Αναλυτικότερα:

1. **Exception handling**: το catch block δεν είναι ποτέ άδειο και ταυτόχρονα δεν χρησιμοποιείται το γενικότερο Exception, αλλά πιο εξειδικευμένα exceptions ανάλογα με το περιεχόμενο του try block.
2. **Imports**: Αναφέρεται κάθε κλάση που γίνεται import ανεξάρτητα από το αν ανήκει στο ίδιο πακέτο με μια άλλη κλάση που έχει ήδη εισαχθεί. Αρχικά δηλώνονται τα android imports, στη συνέχεια εκείνα που έρχονται από τρίτες πηγές και τέλος τα java ή javax imports. Με αυτό τον τρόπο είναι ευκολότερο να κατανοήσει κανείς ποιες κλάσεις χρησιμοποιούνται μέσα στον κώδικα.
3. **Comments**: Στην αρχή κάθε αρχείου καταγράφονται τα δικαιώματα πνευματικής ιδιοκτησίας και για κάθε κλάση ή μέθοδο υπάρχουν Javadoc comments [32] στα οποία περιγράφεται η λειτουργία που προσφέρουν.
4. **Naming Conventions**:
    a. Κάθε μεταβλητή που δεν είναι public static ξεκινάει με m
    b. Κάθε μεταβλητή που είναι static ξεκινάει με s
    c. Κάθε μεταβλητή που είναι public static final, δηλαδή σταθερά, περιέχει μόνο κεφαλαία γράμματα και το σύμβολο _ σε περίπτωση που το όνομα αποτελείται από περισσότερες από μία λέξεις
    d. Όλες οι μεταβλητές με εξαίρεση αυτές που αναφέρονται στο 4.c ονομάζονται με βάση το camelCase στιλ

4.3 Κώδικας

4.3.1 Επικοινωνία Android Studio με Firebase Authentication

Για να χρησιμοποιηθεί η εφαρμογή, είναι απαραίτητη η δημιουργία λογαριασμού ή η σύνδεση σε έναν ήδη υπάρχον. Όταν ο χρήστης κάνει Sign up, αποθηκεύεται στο Firebase το email και ο κωδικός που εισήγαγε, έτσι ώστε να μπορεί να ξανασυνδεθεί με αυτά τα στοιχεία σε περίπτωση αποσύνδεσής του και να έχει πρόσβαση στα δεδομένα που δημιούργησε. Τη στιγμή, επομένως που ο χρήστης πατάει το κουμπί Sign up, καλείται η παρακάτω μέθοδος.

```
private void signUp() {
        private FirebaseAuth mAuth;
        String email = mEmailField.getText().toString();
        String password = mPasswordField.getText().toString();

        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                        Log.d(TAG, "createUser:onComplete:" +
task.isSuccessful());
                        hideProgressDialog();

                        if (task.isSuccessful()) {
                            onAuthSuccess(task.getResult().getUser());
                        } else {
                            Toast.makeText(SignInActivity.this, "Sign Up
Failed",
                                    Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }
```

Χρησιμοποιείται η createUserWithEmailAndPassword μέθοδος της FirebaseAuth κλάσης που δημιουργεί έναν χρήστη με το email και τον κωδικό που δόθηκε.

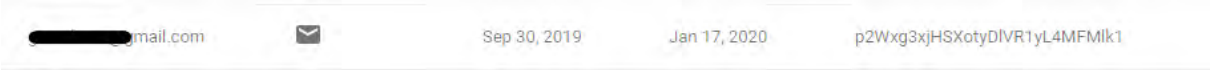


Figure 3. Παράδειγμα χρήστη

Στη συνέχεια καλεί την μέθοδο onAuthSuccess, που δημιουργεί ένα username για τον νέο χρήστη και δημιουργεί μια νέα εγγραφή User η οποία αποθηκεύεται στη βάση δεδομένων και έχει ένα μοναδικό αναγνωριστικό, το οποίο χρησιμοποιείται για να εξακριβώνεται ποια δεδομένα ανήκουν στον συγκεκριμένο χρήστη. Μετά την επιτυχημένη αυτή αποθήκευση, δίνει πρόσβαση στις κύριες λειτουργίες της εφαρμογής.

```
private void onAuthSuccess(FirebaseUser user) {
   String username = usernameFromEmail(user.getEmail());
   User user = new User(name, email);
   mDatabase.child("users").child(userId).setValue(user);

   // Go to MainActivity
   startActivity(new Intent(SignInActivity.this, MainActivity.class));
   finish();
}
```

Σε περίπτωση που ο χρήστης έχει ήδη λογαριασμό στην εφαρμογή, πατάει το κουμπί Sign In και παρόμοια με την μέθοδο signUp, καλείται η signIn, η οποία χρησιμοποιεί την signInWithEmailAndPassword μέθοδο της κλάσης FirebaseAuth. Αν τα στοιχεία είναι σωστά, δίνεται πρόσβαση στις βασικές λειτουργίες της εφαρμογής.

```
private void signIn() {
   String email = mEmailField.getText().toString();
   String password = mPasswordField.getText().toString();

   mAuth.signInWithEmailAndPassword(email, password)
           .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
               @Override
               public void onComplete(@NonNull Task<AuthResult> task) {
                   Log.d(TAG, "signIn:onComplete:" +
task.isSuccessful());
                   hideProgressDialog();

                   if (task.isSuccessful()) {
                       startActivity(new Intent(SignInActivity.this,
MainActivity.class));
                       finish();
                   } else {
                       Toast.makeText(SignInActivity.this, "Sign In
Failed",
                               Toast.LENGTH_SHORT).show();
                   }
               }
           });
}
```

Τέλος, αν ο χρήστης επιθυμεί να αποσυνδεθεί από την εφαρμογή, τότε διαλέγει από το μενού την επιλογή Log out και καλείται το παρακάτω κομμάτι κώδικα, το οποίο χρησιμοποιεί και πάλι την κλάση FirebaseAuth και τις δυνατότητές της. Μετά την αποσύνδεση, η εφαρμογή επιστρέφει στην αρχική σελίδα σύνδεσης ή δημιουργίας νέου λογαριασμού.

```
FirebaseAuth.getInstance().signOut();
startActivity(new Intent(this, SignInActivity.class));
finish();
```

4.3.2 Επικοινωνία Android Studio με Firebase Realtime Database

Σε περίπτωση που ο χρήστης επιθυμεί να δημιουργήσει μια νέα εγγραφή στους προϋπολογισμούς του, παρουσιάζεται παρακάτω πώς το Android Studio στέλνει αυτή την πληροφορία στην βάση δεδομένων.

Αρχικά, η κλάση Budget, αντιπροσωπεύει τον προϋπολογισμό και περιέχει όλες τις σχετικές με αυτό πληροφορίες. Η μέθοδος toMap(), ουσιαστικά παίρνει το περιεχόμενο των μεταβλητών και το μετατρέπει σε JSON, που είναι και η μορφή στην οποία θέλουμε να αποθηκεύσουμε την εγγραφή στη βάση δεδομένων.

```
public class Budget {

   public String uid;
   public String fromDate;
   public String toDate;
   public String title;
   public String incomeId;
   public Double amount;
   public Double spentAmount;

   public Budget() {
       // Default constructor required for calls to
DataSnapshot.getValue(Expense.class)
   }

   public Budget(String uid, String fromDate, String toDate, String
title, String relatedIncome, Double amount, Double spentAmount) {
       this.uid = uid;
       this.fromDate = fromDate;
       this.toDate = toDate;
       this.title = title;
       this.incomeId = relatedIncome;
       this.amount = amount;
       this.spentAmount = spentAmount;
   }

   @Exclude
   public Map<String, Object> toMap() {
       HashMap<String, Object> result = new HashMap<>();
```

```
        result.put("uid", uid);
        result.put("fromDate", fromDate);
        result.put("toDate", toDate);
        result.put("title", title);
        result.put("incomeId", incomeId);
        result.put("amount", amount);
        result.put("spentAmount", spentAmount);

        return result;
    }
}
```

Αποθήκευση στη βάση δεδομένων

Όταν ο χρήστης αποθηκεύσει ή τροποποιήσει έναν προϋπολογισμό, καλείται η παρακάτω μέθοδος. Η μεταβλητή mDatabase είναι τύπου DatabaseReference, αποτελεί μια αναφορά στη βάση δεδομένων και χρησιμοποιείται για την ανάγνωση ή επεξεργασία δεδομένων σε εκείνη την τοποθεσία. Αφού, επομένως, η πληροφορία έρθει στην κατάλληλη μορφή, καλείται η updateChildren μέθοδος της κλάσης DatabaseReference και αποθηκεύεται η καινούργια ή τροποποιημένη εγγραφή στο μονοπάτι user-budgets/uid, όπου uid το μοναδικό αναγνωριστικό του χρήστη.

```
private void createNewBudget(String userId, String fromDate, String
toDate, String title, String relatedIncome, Double amount, Double
spentAmount) {
    // Create new expense at /user-posts/$userid/$postid and at
    // /posts/$postid simultaneously
    String key = mDatabase.child("budgets").push().getKey();
    Budget budget = new Budget(userId, fromDate, toDate, title,
relatedIncome,amount, spentAmount);
    Map<String, Object> budgetValues = budget.toMap();

    Map<String, Object> childUpdates = new HashMap<>();
    if (isUpdate) {
        childUpdates.put(mBudgetKey, budgetValues);
        mDatabase.child("user-
budgets").child(mUserKey).updateChildren(childUpdates);
    } else {
        childUpdates.put("/user-budgets/" + userId + "/" + key,
budgetValues);
        mDatabase.updateChildren(childUpdates);
    }
}
```

Ανάκτηση εγγραφών από τη βάση δεδομένων

Για την εμφάνιση των εγγραφών που έχει ήδη δημιουργήσει ο χρήστης, χρησιμοποιείται η κλάση FirebaseRecyclerAdapter, η οποία λειτουργεί σαν τον RecyclerView που περιεγράφηκε στην αντίστοιχη ενότητα, ωστόσο τα δεδομένα που εμφανίζει προέρχονται απευθείας από την Realtime Database του Firebase. Αρχικά, λοιπόν, πρέπει να οριστεί ένα query το οποίο θα ορίζει ποιες εγγραφές θα δείχνει ο RecyclerView. Παρακάτω φαίνεται το query που φέρνει τους προϋπολογισμούς του συγκεκριμένου χρήστη.

```
Query recentPostsQuery = mDatabase.child("user-budgets").child(getUid());
```

Έπειτα το αποτέλεσμα που επιστρέφεται, αναλύεται έτσι ώστε να έχει τη μορφή της Budget κλάσης που έχει οριστεί, έτσι ώστε να είναι δυνατή η ανάκτηση των πεδίων κάθε εγγραφής.

```
FirebaseRecyclerOptions options = new
FirebaseRecyclerOptions.Builder<Budget>()
                .setQuery(budgetsQuery, Budget.class)
                .build();
```

Πλέον μπορεί να οριστεί ο FirebaseRecyclerAdapter όπως παρακάτω.

```
mAdapter = new FirebaseRecyclerAdapter<Budget, BudgetViewHolder>(options)
```

Διαγραφή από τη βάση δεδομένων

Επιπλέον υπάρχει δυνατότητα διαγραφής μια εγγραφής από τον χρήστη. Πιο συγκεκριμένα, αν επιλέξει παρατεταμένα έναν προϋπολογισμό, καλείται η onLongClick και εμφανίζει ένα διάλογο, ο οποίος ζητάει επιβεβαίωση για τη διαγραφή της συγκεκριμένης εγγραφής. Αν ο χρήστης το επιβεβαιώσει, τότε καλείται η μέθοδος removeValue() της DatabaseReference, και ο προϋπολογισμός αφαιρείται από τη βάση δεδομένων.

```
viewHolder.itemView.setOnLongClickListener(new View.OnLongClickListener()
{
   @Override
   public boolean onLongClick(View v) {
       AlertDialog.Builder builder = new
AlertDialog.Builder(BudgetListActivity.this);
       builder.setTitle("Confirm Delete");
       // Add the buttons
       builder.setPositiveButton("Delete", new
DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
```

```
                mDatabase =
FirebaseDatabase.getInstance().getReference().child("user-budgets");

mDatabase.child(getUid()).child(budgetKey).addListenerForSingleValueEvent
(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {

BudgetViewHolder.getAllBudgets().remove(budgetKey);
                        dataSnapshot.getRef().removeValue();

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {

                    }
                });
            }
        });
        builder.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                // User cancelled the dialog
            }
        });
        AlertDialog dialog = builder.create();
        dialog.show();
        return true;
    }
});
```

4.3.3 Υπολογισμός Πορτοφολιού

Για τον υπολογισμό του συνολικού πορτοφολιού δημιουργήθηκε μια στατική μεταβλητή τύπου Double όπως παρακάτω.

```
public static Double totalWallet = 0.0;
```

Στη συνέχεια, για κάθε αντικείμενο του RecyclerView, γίνεται έλεγχος για το αν είναι έσοδο ή έξοδο. Αν ανήκει στην πρώτη κατηγορία, τότε το ποσό του πρέπει να προστεθεί στο

συνολικό πορτοφόλι ενώ αν είναι έξοδο πρέπει να αφαιρεθεί. Ο κώδικας που εκφράζει αυτόν τον υπολογισμό είναι ο παρακάτω.

```
if(expense.isIncome && !allExpenses.containsKey(expenseId)) {
   totalWallet = totalWallet + expense.amount;
   allExpenses.put(expenseId, expense);
} else if(!expense.isIncome && !allExpenses.containsKey(expenseId)){
   totalWallet = totalWallet - expense.amount;
   allExpenses.put(expenseId, expense);
}
```

Είναι επίσης απαραίτητο να ληφθεί υπόψη ότι είναι δυνατή η αλλαγή δεδομένων, επομένως σε περίπτωση που το ποσό μια εγγραφής αλλάξει, η διαφορά προστίθεται ή αφαιρείται από το πορτοφόλι με τη χρήση του παρακάτω κώδικα.

```
if(isIncome){
   PostViewHolder.totalWallet +=
Double.valueOf(mAmountView.getText().toString()) - expense.amount;
}
else {
   PostViewHolder.totalWallet += -
Double.valueOf(mAmountView.getText().toString()) + expense.amount;
}
```

Στην μεταβλητη mAmountView, η οποία είναι Text, ο χρήστης έχει εισάγει το νέο ποσό της εγγραφής ενώ στο πεδίο expense.amount περιέχεται η παλιά του τιμή. Η διαφορά αυτών των δύο ποσών προστίθεται ή αφαιρείται από το συνολικό πορτοφόλι ανάλογα με την τιμή της Boolean μεταβλητής isIncome.

Το ίδιο ισχύει και για τη διαγραφή μια εγγραφής, όπου το ποσό το οποίο της αντιστοιχούσε προστίθεται στο πορτοφόλι, αν ήταν έξοδο, διαφορετικά αφαιρείται.

```
if(expenseToDelete.isIncome) {
   PostViewHolder.totalWallet -= expenseToDelete.amount;
   mWalletAmount.setText(df.format(PostViewHolder.totalWallet));
} else {
   PostViewHolder.totalWallet += expenseToDelete.amount;
   mWalletAmount.setText(df.format(PostViewHolder.totalWallet));
```

4.3.4 Υπολογισμός εναπομείναντος προϋπολογισμού

Εικόνα 25. Εγγραφή προϋπολογισμού

Όταν υπάρχει ένας προϋπολογισμός, τα νέα έξοδα είναι δυνατόν να προστεθούν σε αυτόν, έτσι ώστε να υπάρχει μια ένδειξη ότι έχουν ξοδευτεί κάποια χρήματα που προορίζονταν για αυτόν τον σκοπό. Γι' αυτόν τον λόγο, η πληροφορία για την εγγραφή budget που αποθηκεύεται στη βάση δεδομένων, περιέχει πέρα από το συνολικό ποσό του προϋπολογισμού (amount), και το ποσό που έχει καταναλωθεί (spentAmount). Κάθε φορά, λοιπόν, που προστίθεται ένα έξοδο στο συγκεκριμένο budget, η παρακάτω εντολή αφαιρεί το ποσό από αυτό.

```
budgetToUpdate.spentAmount += expense.amount;
```

Έχοντας ορίσει μια μεταβλητή η οποία αναπαριστά την γραμμή προόδου που φαίνεται στην εικόνα,

```
private ProgressBar mProgressBar;
```

μπορούμε να υπολογίσουμε το ποσοστό του προϋπολογισμού που έχει καταναλωθεί, ως εξής:

```
int percentage = (int) ((budget.spentAmount * 100 )/ budget.amount);
mProgressBar.setProgress(percentage);
```

Όσο πιο μεγάλο είναι το ποσοστό που υπολογίστηκε, τόσο πιο γεμάτη είναι η μπάρα προόδου.

5 Επίλογος

5.1 Πλεονεκτήματα και Μειονεκτήματα

Η συγκεκριμένη εφαρμογή έχει πολλά οφέλη για τον χρήστη. Αρχικά, του δίνει τον πλήρη έλεγχο των εξόδων του, έτσι ώστε να μην σπαταλάει τα χρήματα του χωρίς να το συνειδητοποιεί. Παράλληλα, είναι εύχρηστη, καθώς δεν περιέχει πολύπλοκες λειτουργίες και φορτωμένη διεπαφή. Η βάση δεδομένων που χρησιμοποιείται, σε αντίθεση με άλλους servers, είναι ιδιαίτερα γρήγορη και εμφανίζει απευθείας τις πληροφορίες που χρειάζεται να εμφανιστούν στον χρήστη, ενώ για οποιαδήποτε εισαγωγή, τροποποίηση ή διαγραφή που γίνεται στην εφαρμογή, ενημερώνεται σε μηδενικό χρόνο. Ένα άλλο πλεονέκτημα που προσφέρει μια βάση δεδομένων που βρίσκεται στο cloud είναι ότι ο χρήστης δεν περιορίζεται στην χρήση της εφαρμογής σε μία μόνο συσκευή, αλλά μπορεί να έχει πρόσβαση και σε οποιαδήποτε άλλη, εφόσον θυμάται το email και τον κωδικό πρόσβασης. Σημαντικό είναι το γεγονός ότι για τις απαιτήσεις του συγκεκριμένου project, δεν είναι αναγκαία κάποια συνδρομή επί πληρωμή για την αγορά των υπηρεσιών που χρησιμοποιήθηκαν, η δωρεάν έκδοσή τους επαρκεί και άρα ο προγραμματιστής δεν υποχρεούται να χρεώσει τους χρήστες για να καλύψει αυτό το ποσό. Η εφαρμογή, επομένως, διατίθεται δωρεάν.

Από την άλλη μεριά, η συγκεκριμένη υλοποίηση έχει και κάποιους περιορισμούς. Για αρχή, είναι απαραίτητη η πρόσβαση στο διαδίκτυο, καθώς τα δεδομένα ανακτώνται τη χρονική στιγμή που ο χρήστης ανοίγει την εφαρμογή. Σε αντίθετη περίπτωση, οι εγγραφές δεν εμφανίζονται και η εφαρμογή δεν μπορεί πρακτικά να χρησιμοποιηθεί. Επιπλέον, επειδή χρησιμοποιούνται κάποια αρκετά καινούργια εργαλεία του android, η εφαρμογή δεν είναι συμβατή με πολύ παλιές συσκευές και πιο συγκεκριμένα όσες έχουν έκδοση λειτουργικού παλαιότερη από το Android 6.0. Οι χρήστες με συσκευές που δεν πληρούν τις ελάχιστες προϋποθέσεις για την εγκατάσταση της εφαρμογής αποτελούν περίπου το 25% του συνολικού πληθυσμού που χρησιμοποιεί Android. Τέλος, η εφαρμογή έχει αναπτυχθεί μόνο για περιβάλλον Android και έτσι δεν μπορεί να εγκατασταθεί σε iPhone. Για την υλοποίηση της σε iOS θα έπρεπε να χρησιμοποιηθούν άλλα εργαλεία με τα οποία ο προγραμματιστής δεν έχει εξοικείωση, ενώ δεν είναι διαθέσιμη κάποια συσκευή iPhone έτσι ώστε να γίνει ο έλεγχος για τη σωστή λειτουργία της εφαρμογής.

5.2 Μελλοντικές προσθήκες και νέες τεχνολογίες

Παρά τους όποιους περιορισμούς, η εφαρμογή μπορεί να εξελιχθεί περεταίρω και να γίνει ακόμα πιο εύχρηστη.

Μια ενδιαφέρουσα μελλοντική προσθήκη είναι η εγγραφή στην εφαρμογή με τη χρήση λογαριασμού Google ή κάποιου μέσου κοινωνικής δικτύωσης όπως το Facebook. Μαζί με αυτήν, θα προστεθεί το προφίλ του χρήστη στις επιλογές του κύριου μενού. Από εκεί, θα είναι δυνατή η επεξεργασία του ονόματος, του τρόπου σύνδεσης και των διαθέσιμων κατηγοριών που εμφανίζονται στα έξοδα και τα έσοδα. Επιπλέον, ένα χαρακτηριστικό παρόμοιων εφαρμογών που περιεγράφηκαν παραπάνω, είναι η δυνατότητα σύνδεσης τραπεζικού λογαριασμού. Κάτι τέτοιο θα μπορούσε να μελετηθεί, ωστόσο η υλοποίησή του γίνεται πολυπλοκότερη καθώς απαιτείται επικοινωνία με τα APIs της εκάστοτε ελληνικής τράπεζας.

Όσον αφορά τις τεχνολογίες που μπορούν να χρησιμοποιηθούν, θα μπορούσε να γίνει μετατροπή του κώδικα της εφαρμογής από Java σε Kotlin, καθώς η επικοινωνία με το Firebase φαίνεται να είναι ευκολότερη. Επίσης, το Cloud Firestore είναι δυνατό να αντικαταστήσει την Real Time Database, έτσι ώστε να είναι λιγότερο περίπλοκη η αναζήτηση εγγραφών που απαιτούν πολλά φίλτρα ή έχουν πολύπλοκη ιεραρχία. Τέλος, το Google Cloud Platform προσφέρει υπηρεσίες analytics έτσι ώστε να γίνεται ανάλυση των δεδομένων που δημιουργεί ο χρήστης και να παρουσιάζονται περιγραφικά και περιεκτικά γραφήματα σχετικά με τις ενέργειες του. Είναι σημαντικό, ωστόσο, να σημειωθεί ότι τέτοιου είδους υπηρεσίες είναι συνήθως επί πληρωμή.

Βιβλιογραφία

[1] «Android (operating system),» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Android_(operating_system).

[2] «Platform Architecture,» [Ηλεκτρονικό]. Available: https://developer.android.com/guide/platform.

[3] N. Gove, «How garbage collection works in android,» 1 January 2016. [Ηλεκτρονικό]. Available: https://medium.com/@nitinkumargove/how-garbage-collection-works-in-art-in-android-5e7b1a5f0ed4.

[4] M. Rouse, «Native code,» [Ηλεκτρονικό]. Available: https://searchapparchitecture.techtarget.com/definition/native-code.

[5] «Distribution dashboard,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/dashboards.

[6] «Support different platform versions,» [Ηλεκτρονικό]. Available: https://developer.android.com/training/basics/supporting-devices/platforms.html.

[7] «Understand the Activity Lifecycle,» [Ηλεκτρονικό]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle.

[8] «Fragments,» [Ηλεκτρονικό]. Available: https://developer.android.com/guide/components/fragments.

[9] E. Rosenberg, «The 8 Best Budgeting Apps of 2020,» 15 January 2020. [Ηλεκτρονικό]. Available: https://www.thebalance.com/best-budgeting-apps-4159414.

[10] E. Hong, «The 5 Best Budgeting Apps,» 19 August 2019. [Ηλεκτρονικό]. Available: https://www.investopedia.com/personal-finance/personal-finance-apps/.

[11] N. Marceau, «Spendee Review: The Pros & Cons of This Expense-Tracking App,» 10 March 2019. [Ηλεκτρονικό]. Available: https://www.frugalforless.com/spendee-review-the-pros-cons-of-this-expense-tracking-app/.

[12] «Kotlin (programming language),» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Kotlin_(programming_language).

[13] «Τι είναι το cloud computing;,» 15 October 2017. [Ηλεκτρονικό]. Available: https://www.csc.com.gr/cloud-computing/.

[14] «Google Cloud Platform Overview,» [Ηλεκτρονικό]. Available: https://cloud.google.com/docs/overview/.

[15] C. Gehman, «What is Version Control,» 23 May 2019. [Ηλεκτρονικό]. Available: https://www.perforce.com/blog/vcs/what-is-version-control.

[16] «Version control,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Version_control.

[17] «Fork,» [Ηλεκτρονικό]. Available: https://git-fork.com/.

[18] C. Esplin, «What is Firebase?,» 24 October 2016. [Ηλεκτρονικό]. Available: https://howtofirebase.com/what-is-firebase-fcb8614ba442.

[19] «Firebase Authentication,» [Ηλεκτρονικό]. Available: https://firebase.google.com/docs/auth.

[20] «Choose a Database: Cloud Firestore or Realtime Database,» [Ηλεκτρονικό]. Available: https://firebase.google.com/docs/database/rtdb-vs-firestore.

[21] «Hatchful,» [Ηλεκτρονικό]. Available: https://hatchful.shopify.com/.

[22] «Create a List with RecyclerView,» [Ηλεκτρονικό]. Available: https://developer.android.com/guide/topics/ui/layout/recyclerview.

[23] M. Aravind, «What is the difference between ListView and RecyclerView?,» 30 December 2017. [Ηλεκτρονικό]. Available: https://medium.com/@manuaravindpta/what-is-the-difference-between-listview-and-recyclerview-bcd82c64ffbb.

[24] «Cards,» [Ηλεκτρονικό]. Available: https://material.io/components/cards/.

[25] «Create an Android project,» [Ηλεκτρονικό]. Available: https://developer.android.com/training/basics/firstapp/creating-project.

[26] «Sign your app,» [Ηλεκτρονικό]. Available: https://developer.android.com/studio/publish/app-signing.html.

[27] «SHA-1,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/SHA-1.

[28] M. Anicas, «An Introduction to OAuth 2,» 21 July 2014. [Ηλεκτρονικό]. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2.

[29] G. Yadav, «Get SHA1 from Android Studio,» 13 October 2017. [Ηλεκτρονικό]. Available: https://medium.com/pen-bold-kiln-press/sha-1-android-studio-ec02fb893e72.

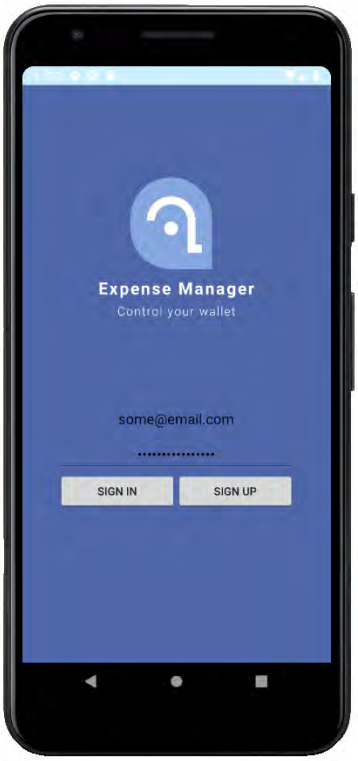
[30] «Add Firebase to your Android project,» [Ηλεκτρονικό]. Available: https://firebase.google.com/docs/android/setup.

[31] «AOSP Java Code Style for Contributors,» [Ηλεκτρονικό]. Available: https://source.android.com/setup/contribute/code-style.

[32] «How to Write Doc Comments for the Javadoc Tool,» [Ηλεκτρονικό]. Available: https://www.oracle.com/technetwork/java/javase/documentation/index-137868.html.

Παράρτημα

Παράδειγμα Χρήσης



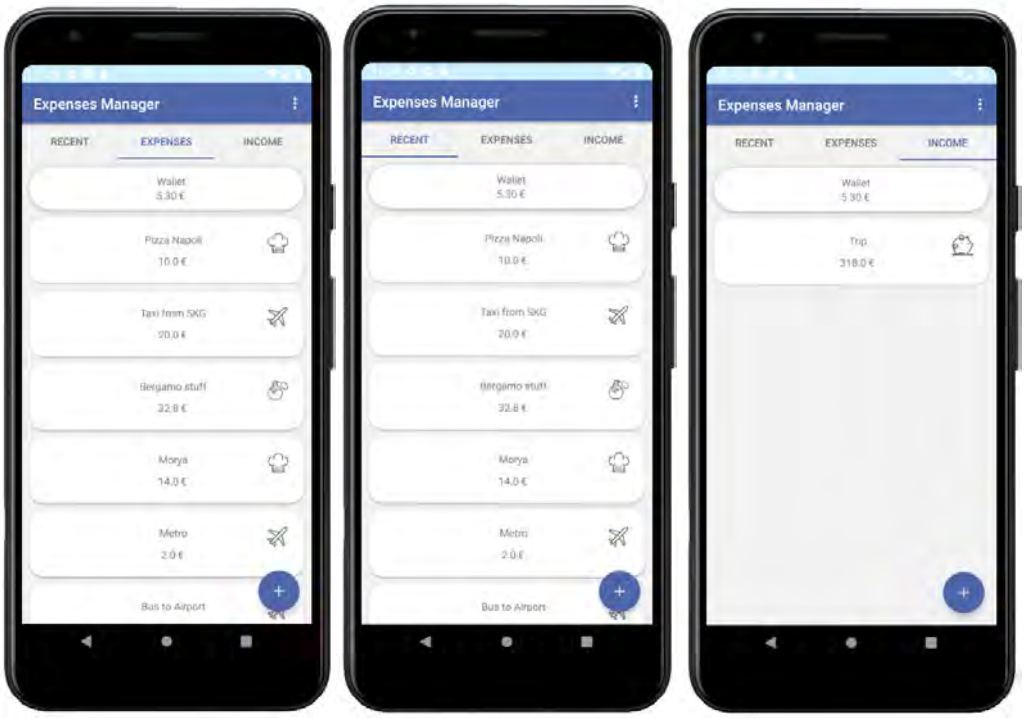
Στην πρώτη οθόνη που εμφανίζεται κατά την είσοδο του χρήστη στην εφαρμογή, ζητείται ένα email και ένας κωδικός πρόσβασης. Στην περίπτωση που δεν είναι εγγεγραμμένος χρήστης, τα στοιχεία που δίνει αποθηκεύονται στη βάση έτσι ώστε να δημιουργηθεί λογαριασμός για εκείνον.

Εικόνα 26. Login στην εφαρμογή

Μετά την επιτυχημένη είσοδο στην εφαρμογή, εμφανίζονται η κύρια οθόνη, που περιέχει τρεις καρτέλες. Αυτή που εμφανίζεται πρώτη είναι η καρτέλα Recent και περιέχει τα πιο πρόσφατα έσοδα και έξοδα που έχει κάνει ο χρήστης.

Κάτω από τις καρτέλες, υπάρχει ένα εικονικό πορτοφόλι του χρήστη, το οποίο δείχνει πόσα χρήματα του απομένουν με βάση τις εγγραφές που έχει εισάγει. Πιο συγκεκριμένα, αν προστεθεί νέο εισόδημα, τότε αυτό το ποσό προστίθεται στο πορτοφόλι και αντίστοιχα αν καταγραφούν έξοδα, αφαιρούνται από το πορτοφόλι.

Στην καρτέλα Expenses εμφανίζονται μόνο τα έξοδα και στην καρτέλα Income μόνο τα έσοδα.



Εικόνα 27. Έξοδα χρήστη *Εικόνα 28. Πρόσφατες εγγραφές χρήστη* *Εικόνα 29. Έσοδα χρήστη*

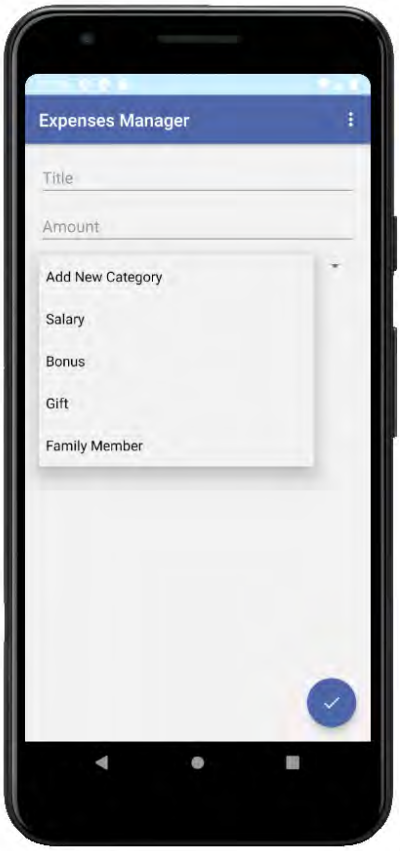
Εικόνα 30. Προσθήκη νέου εσόδου

Έστω ότι ο χρήστης κέρδισε δύο χιλιάδες ευρώ στο λαχείο και επιθυμεί να δημιουργήσει ένα καινούργιο έσοδο για να καταγράψει αυτή του την επιτυχία. Επιλέγοντας το κουμπί κάτω δεξιά στην καρτέλα των εσόδων, του εμφανίζεται μια φόρμα που ζητάει ένα όνομα, το ποσό που του αναλογεί και δίνει τη δυνατότητα επιλογής ημερομηνίας και κατηγορίας.

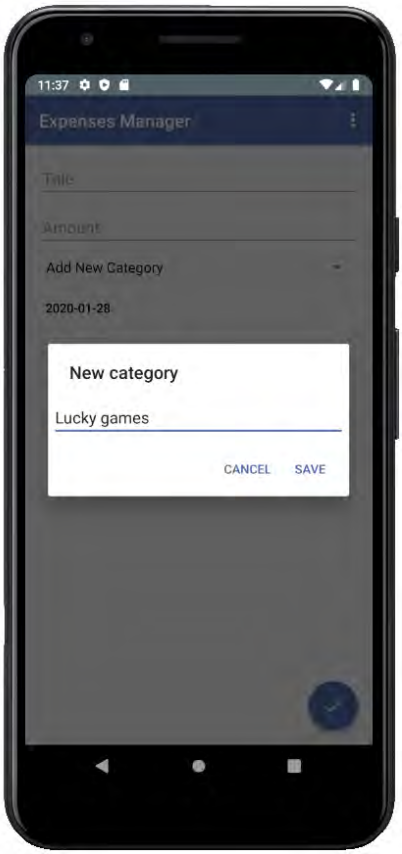
Για την επιλογή κατηγορίας υπάρχουν κάποιες προϋπάρχουσες επιλογές αλλά είναι δυνατό να δημιουργήσει ο χρήστης τη δική του κατηγορία.

Επιλέγοντας τη δημιουργία νέας κατηγορίας, εμφανίζεται ένας διάλογος που ζητάει το όνομα που ο χρήστης επιθυμεί να της δώσει.

Μετά την αποθήκευση της νέας κατηγορίας, το πεδίο συμπληρώνεται με αυτή την επιλογή.



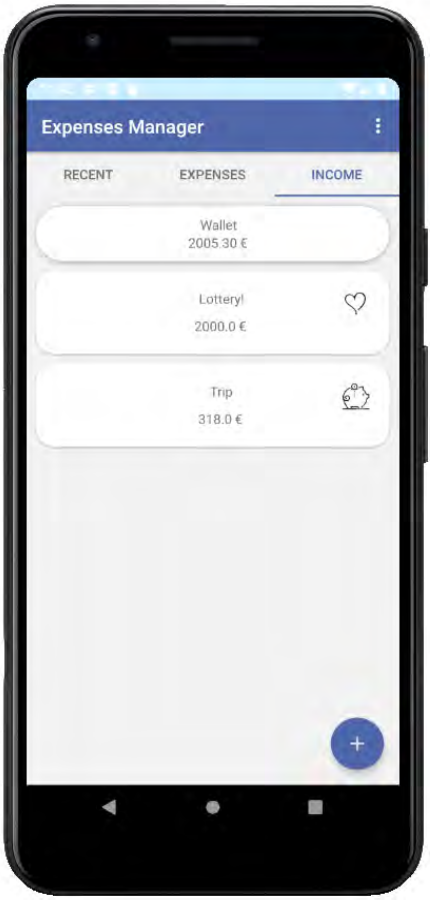
Εικόνα 31. Επιλογή κατηγορίας



Εικόνα 32. Δημιουργία νέας κατηγορίας

Όταν ο χρήστης συμπληρώσει όλα τα πεδία και πατήσει το κουμπί κάτω δεξιά, η εγγραφή αποθηκεύεται στη βάση και εμφανίζεται στη λίστα με όλα τα εισοδήματα.

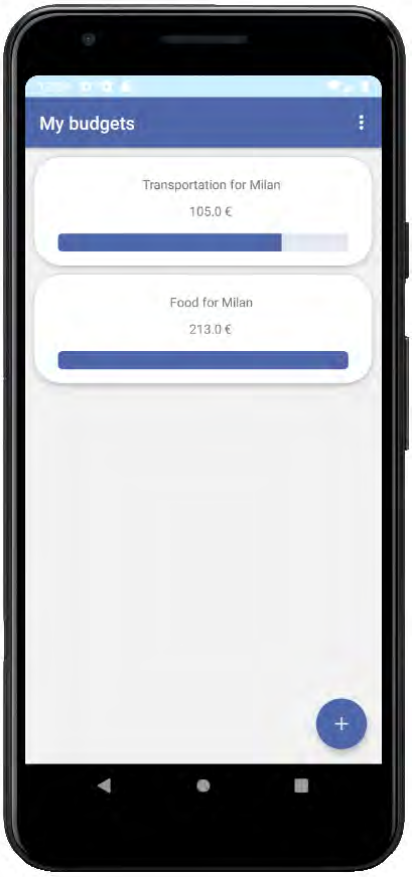
Εικόνα 33. Λεπτομέρειες της εγγραφής λαχείου



Εικόνα 34. Εμφάνιση νέας εγγραφής στα έσοδα

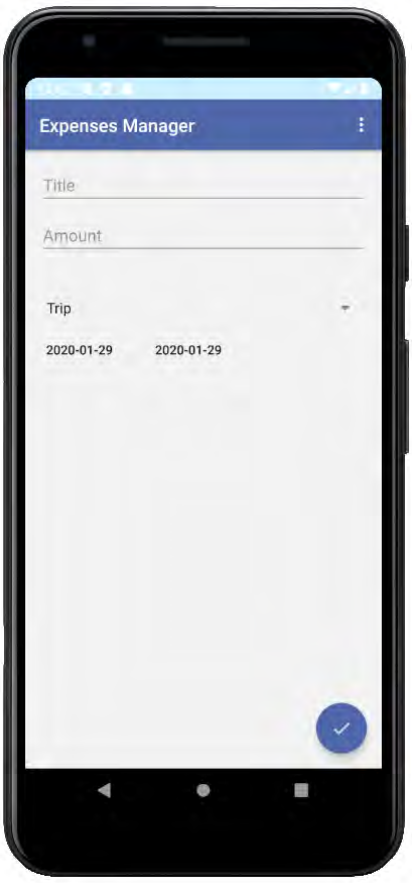
Στη συνέχεια ο χρήστης θέλει να δημιουργήσει έναν προϋπολογισμό βασισμένο στο παραπάνω έσοδο του, με σκοπό να οργανώσει την κατανομή αυτών των χρημάτων. Έχει σκοπό να ξοδέψει κάποιο ποσό του λαχείου σε ένα ταξίδι. Στο μενού πάνω δεξιά επιλέγει My Budgets και εμφανίζεται η οθόνη με όλους τους προϋπολογισμούς του. Πατώντας το κουμπί προσθήκης κάτω δεξιά, ανοίγει η φόρμα που πρέπει να συμπληρώσει για να αποθηκεύσει τη νέα εγγραφή.

Εικόνα 35. Εμφάνιση μενού

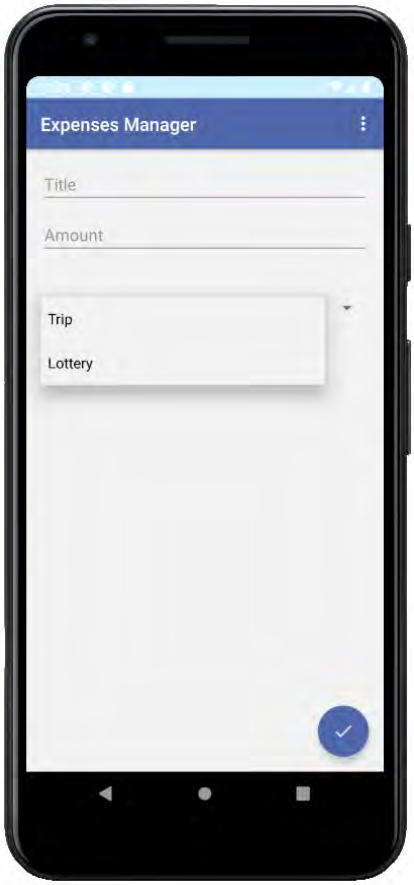


Εικόνα 36. Προβολή όλων των προϋπολογισμών

Η φόρμα είναι παρόμοια με εκείνη των εσόδων μόνο που αυτή τη φορά ο χρήστης καλείται να επιλέξει ένα από τα υπάρχοντα εισοδήματα, έτσι ώστε ο προϋπολογισμός να συνδέεται με κάποιο από αυτά.

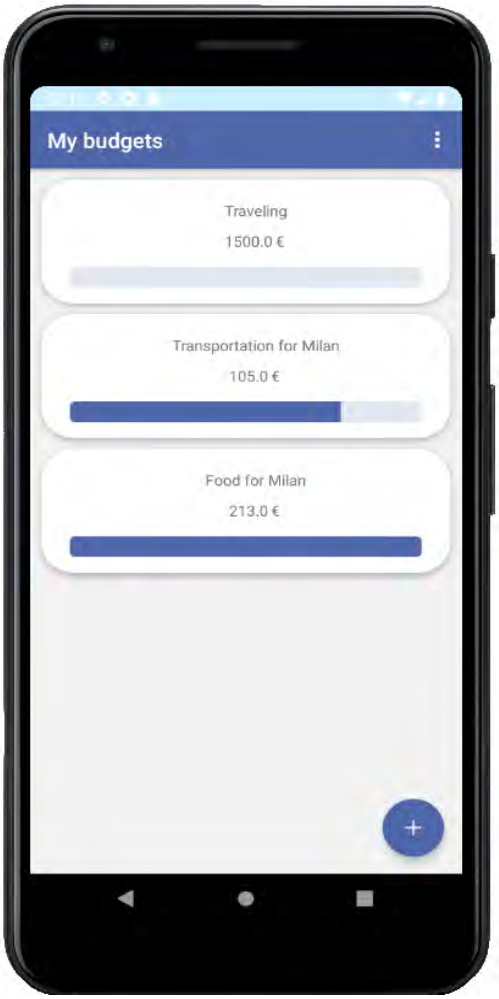


Εικόνα 37. Νέος προϋπολογισμός

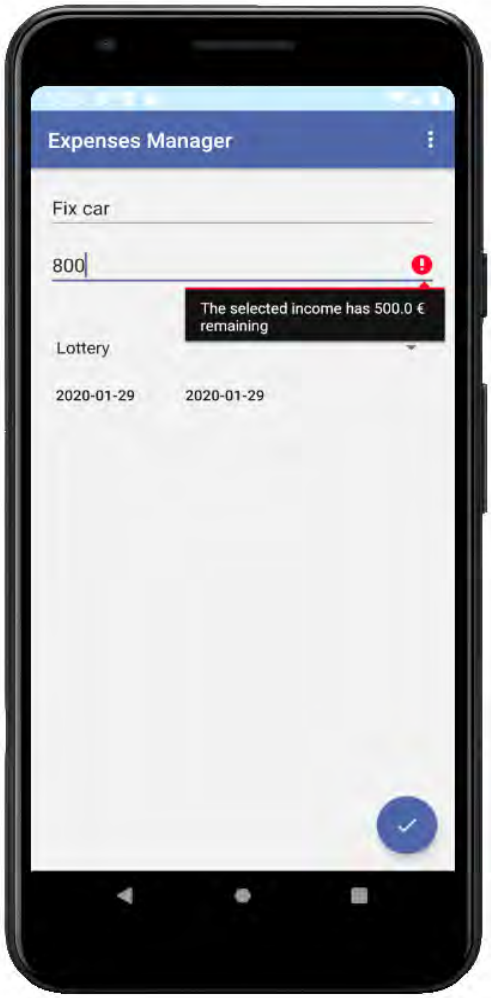


Εικόνα 38. Εμφάνιση διαθέσιμων εσόδων

Πατώντας το κουμπί κάτω δεξιά η εφαρμογή επιστρέφει στην οθόνη με όλους τους προϋπολογισμούς. Τελικά, ο χρήστης αφού δημιουργήσει έναν προϋπολογισμό χιλίων ευρώ για το ταξίδι του και έναν αξίας πεντακοσίων ευρώ που σκοπεύει να ξοδέψει σε διάφορες αγορές, σκέφτεται ότι θέλει να κάνει διάφορες επιδιορθώσεις στο αμάξι του που πιθανότατα να του κοστίσουν οκτακόσια ευρώ.



Εικόνα 40. Νέα εγγραφή προϋπολογισμού



Εικόνα 39. Μήνυμα λάθους

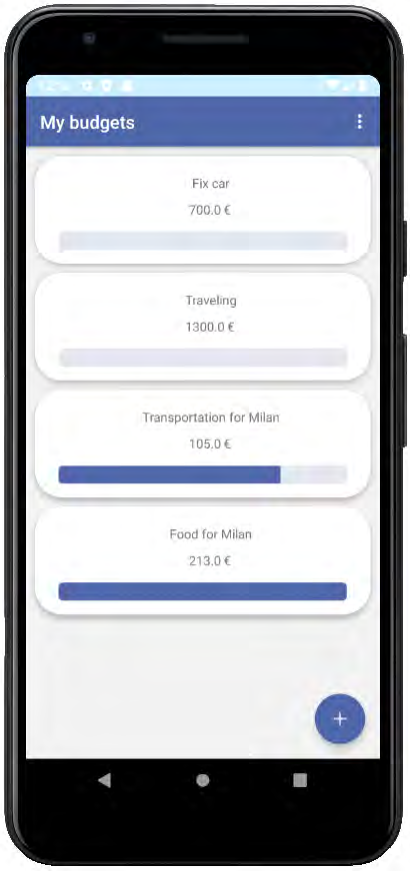
Εισάγει λοιπόν ένα ακόμη προϋπολογισμό βασισμένο στο λαχείο που κέρδισε, ωστόσο κατά την αποθήκευσή του, του εμφανίζεται το μήνυμα ‘The selected income has 500€ remaining’.

Αυτό το μήνυμα εμφανίζεται επειδή οι δύο προϋπολογισμοί έχουν άθροισμα δύο χιλιάδες τριακόσια ευρώ και άρα ξεπερνούν το ποσό του λαχείου που ήταν δύο χιλιάδες ευρώ.

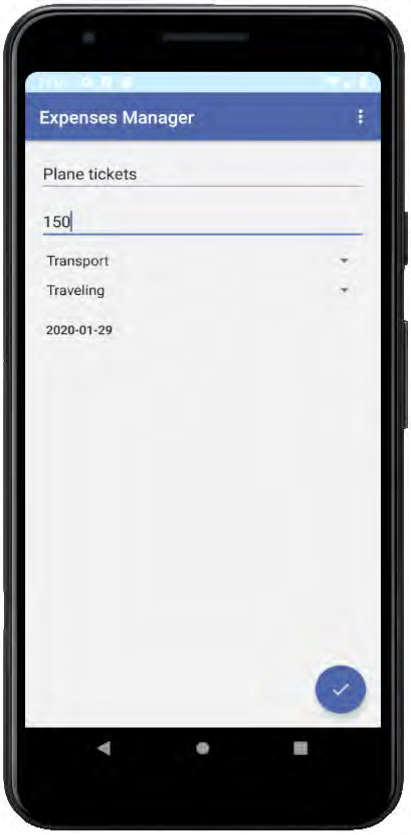
Πρέπει επομένως ο χρήστης να κάνει κάποιες αλλαγές στους προϋπολογισμούς του έτσι ώστε το άθροισμά τους να είναι μικρότερο ή ίσο του εισοδήματος.

Τελικά αποφασίζει να μειώσει το αρχικό ποσό που σκόπευε να ξοδέψει στο ταξίδι, έτσι ώστε να απομείνουν περισσότερα χρήματα για την επισκευή του αμαξιού. Επομένως, ενημερώνει τον προϋπολογισμό Traveling με ποσό από τα χίλια πεντακόσια στα χίλια τριακόσια ευρώ και δημιουργεί έναν νέο προϋπολογισμό Fix car ύψους επτακοσίων ευρώ.

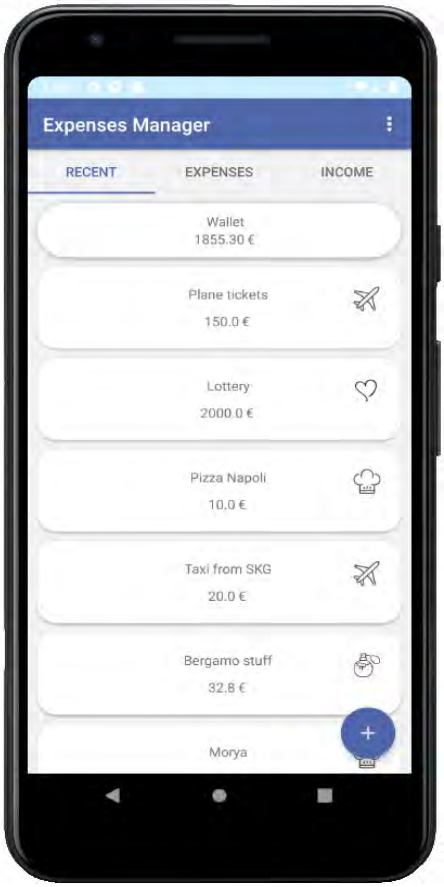
Κάποιες μέρες μετά ο χρήστης αγοράζει τα αεροπορικά εισιτήρια για το ταξίδι του και θέλει να καταγράψει αυτό το έξοδο. Πατώντας το κουμπί προσθήκης κάτω δεξιά στην καρτέλα με τις πρόσφατες εγγραφές ή με τα έξοδα, του εμφανίζεται η φόρμα που πρέπει να συμπληρώσει, η οποία είναι ίδια με την αντίστοιχη των εσόδων με τη διαφορά ότι υπάρχει ακόμα ένα πεδίο, αυτό του προϋπολογισμού. Από προεπιλογή, κανένας προϋπολογισμός δεν είναι επιλεγμένος καθώς ο χρήστης πιθανώς να μη θέλει να τον αφαιρέσει από κάποιον από τους υπάρχοντες.



Εικόνα 41. Λίστα με τους νέους προϋπολογισμούς

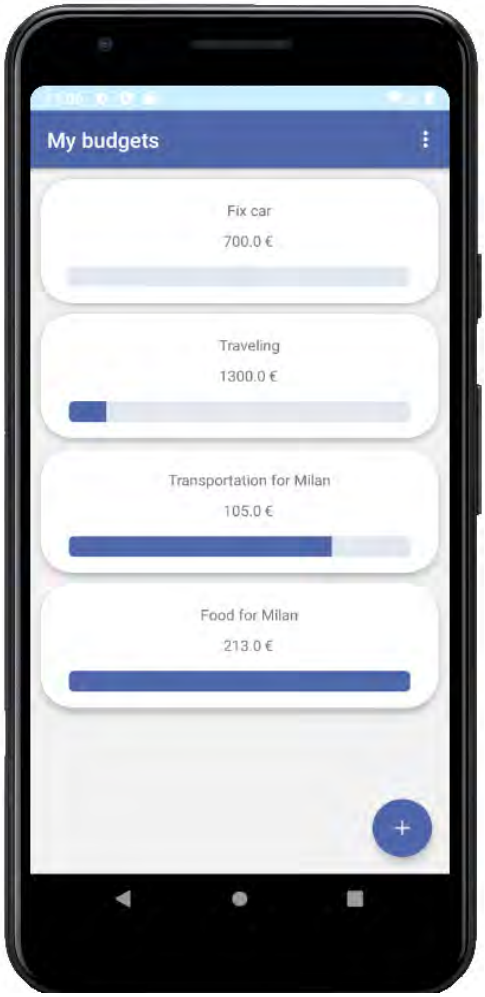


Εικόνα 42. Λεπτομέρειες εξόδου

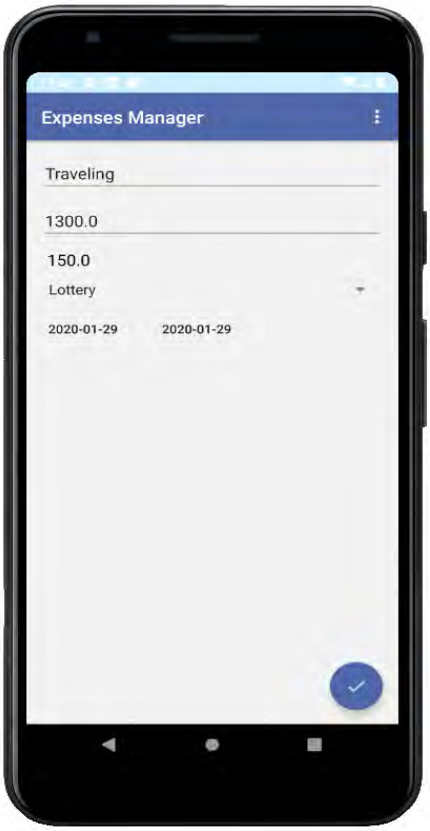


Εικόνα 42. Εμφάνιση νέου εξόδου στη λίστα

Στο συγκεκριμένο παράδειγμα, ωστόσο, θέλει να αφαιρέσει το ποσό των εισιτηρίων από τον προϋπολογισμό που αντιστοιχεί στο ταξίδι του. Αποθηκεύοντας το, το ποσό αυτό αφαιρείται αυτόματα από το budget.

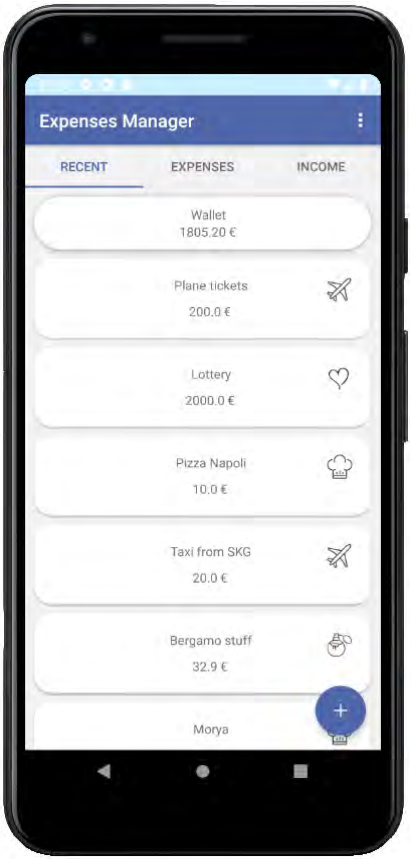


Εικόνα 44. Αλλαγή εναπομείναντος ποσού στον προϋπολογισμό

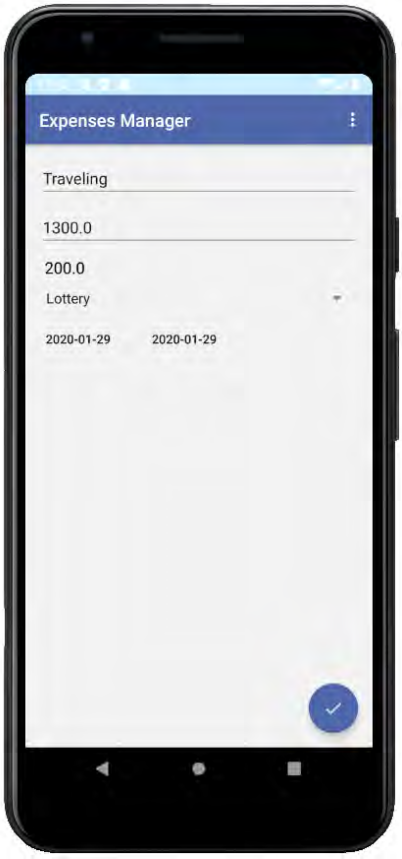


Σε περίπτωση προσθήκης κι άλλου εξόδου ή τροποποίησης κάποιου υπάρχοντος, ενημερώνεται ταυτόχρονα και το σχετικό budget. Έτσι αν αλλάξει η τιμή των εισιτηρίων από εκατόν πενήντα ευρώ σε διακόσια, το ποσό που έχει ξοδευτεί από το budget ενημερώνεται και αυτό.

Εικόνα 43. Λεπτομέρειες ενημερωμένου προϋπολογισμού

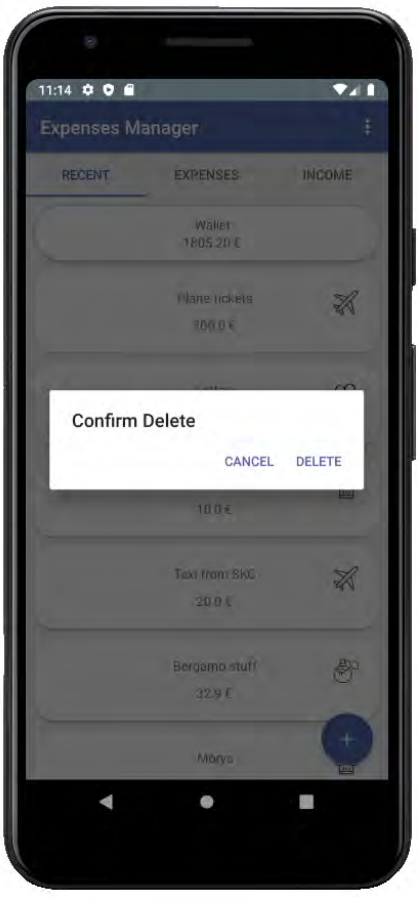


Εικόνα 44. Αλλαγή τιμής εξόδου

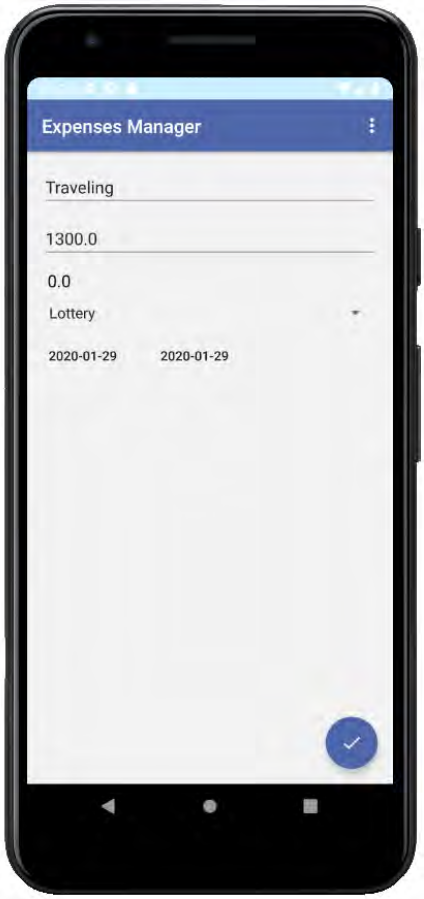


Εικόνα 45. Ενημέρωση προϋπολογισμού

Τέλος, αν ο χρήστης για οποιοδήποτε λόγο θέλει να διαγράψει την αγορά των αεροπορικών ταξιδιών,, πρέπει να πατήσει παρατεταμένα επάνω στην εγγραφή. Τότε εμφανίζεται ένας διάλογος για την επιβεβαίωση της διαγραφής. Αφού ο χρήστης τη διαγράψει, αφαιρείται από το budget Traveling το ποσό που αντιστοιχούσε σε αυτό το έξοδο.



Εικόνα 46. Διαγραφή εγγραφής



Εικόνα 47. Αφαίρεση εξόδου από τον προϋπολογισμό